Intel® DPC++ Compatibility Tool Developer Guide and Reference

Contents

Chapter 1: Intel® DPC++ Compatibility Tool Developer Guide and Reference

Tool Setup and Basic Use 6
Migration 9
Migration Workflow Guidelines 10
Generate a Compilation Database 17
Incremental Migration 20
Migration Rules 22
Migrate a CMake Build Script 26
Debug Migrated Code: Runtime Behavior 28
Query CUDA* to SYCL* API Mapping 34
Analysis Mode 35
Reference 36
Command Line Options Reference 36
Basic Migration Options 37
Advanced Migration Options 39
Code Generation Options 42
Report Generation Options 43
Build Script Options 44
Query API Mapping, Tool Warning, and Help Options 44
intercept-build Options 45
Deprecated Options 46
Alphabetical Option List 46
Diagnostics Reference 52
DPCT1000 59
DPCT1001 60
DPCT1002 61
DPCT1003 62
DPCT1004 63
DPCT1005 63
DPCT1006 64
DPCT1007 66
DPCT1008 66
DPCT1009 68
DPCT1010 69
DPCT1011 70
DPCT1012 72
DPCT1013 74
DPCT1014 74
DPCT1015 75
DPCT1016 76
DPCT1017 77
DPCT1018 78
DPCT1019 80
DPCT1020 81
DPCT1021 83
DPCT1022 83
DPCT1023 84

DPCT1024 85
DPCT1025 87
DPCT1026 87
DPCT1027 90
DPCT1028 93
DPCT1029 94
DPCT1030 94
DPCT1031 95
DPCT1032 95
DPCT1033 96
DPCT1034 97
DPCT1035 98
DPCT1036 99
DPCT1037 100
DPCT1038 100
DPCT1039 102
DPCT1040 103
DPCT1041 104
DPCT1042 106
DPCT1043 109
DPCT1044 110
DPCT1045 110
DPCT1046 113
DPCT1047 115
DPCT1048 116
DPCT1049 117
DPCT1050 118
DPCT1051 120
DPCT1052 121
DPCT1053 122
DPCT1054 122
DPCT1055 124
DPCT1056 125
DPCT1057 126
DPCT1058 128
DPCT1059 129
DPCT1060 130
DPCT1061 131
DPCT1062 132
DPCT1063 133
DPCT1064 134
DPCT1065 135
DPCT1066 136
DPCT1067 137
DPCT1068 137
DPCT1069 138
DPCT1070 140
DPCT1071 141
DPCT1072 141
DPCT1073 142
DPCT1074 144
DPCT1075 145
DPCT1076 146
DPCT1077 147
DPCT1078 148

DPCT1079..... 149
DPCT1080..... 150
DPCT1081..... 151
DPCT1082..... 152
DPCT1083..... 153
DPCT1084..... 154
DPCT1085..... 157
DPCT1086..... 159
DPCT1087 [UPDATE] 160
DPCT1088..... 162
DPCT1089..... 164
DPCT1090..... 165
DPCT1091..... 166
DPCT1092..... 168
DPCT1093..... 168
DPCT1094..... 169
DPCT1095..... 170
DPCT1096..... 170
DPCT1097..... 172
DPCT1098..... 173
DPCT1099..... 174
DPCT1100..... 176
DPCT1101..... 178
DPCT1102..... 179
DPCT1103..... 181
DPCT1104..... 182
DPCT1105..... 183
DPCT1106..... 184
DPCT1107..... 185
DPCT1108..... 185
DPCT1109..... 187
DPCT1110..... 188
DPCT1111..... 189
DPCT1112..... 190
DPCT1113..... 191
DPCT1114..... 193
DPCT1115..... 195
DPCT1116..... 196
DPCT1117..... 197
DPCT1118..... 199
DPCT1119..... 201
DPCT1120..... 201
DPCT1121..... 204
DPCT1122..... 205
DPCT1123..... 205
DPCT1124..... 207
DPCT1125..... 208
DPCT1126..... 210
DPCT1127..... 210
DPCT1128..... 211
DPCT1129..... 213
DPCT1130..... 213
DPCT1131..... 215
DPCT1132..... 215
DPCT2001..... 217

DPCT3000 218
DPCT3001 218
DPCT3002 220
DPCT Namespace Reference 221
CUDA* and SYCL* Programming Model Comparison 228
CUDA* to SYCL* Term Mapping Quick Reference 237
CUDA API Migration Support 240
Frequently Asked Questions 423
Notices and Disclaimers 430

Intel® DPC++ Compatibility Tool Developer Guide and Reference

1

This guide provides a detailed overview of Intel® DPC++ Compatibility Tool features, workflow, and use.

Use Intel® DPC++ Compatibility Tool to help port CUDA* language kernels and library API calls to Data Parallel C++ (DPC++) code.

Related Information

- Intel® DPC++ Compatibility Tool official page
- Intel® DPC++ Compatibility Tool Release Notes
- Install Intel® oneAPI Toolkits and Components
- Get Started with the Intel® DPC++ Compatibility Tool

Tool Setup and Basic Use

Before You Begin

Install and set up Intel® DPC++ Compatibility Tool.

Refer to Get Started with the Intel® DPC++ Compatibility Tool for installation and setup information.

Basic Use

You can run Intel® DPC++ Compatibility Tool from the command line and provide migration instructions using the tool's command-line options. The general command syntax is:

```
dpct [options] [<source0>... <sourceN>]
```

> **NOTE** `c2s` is an alias to the `dpct` command and may be used in its place.

If no directory or file is specified for migration, the tool will try to migrate source files found in the current directory. The default output directory is `dpct_output`. Use the `--out-root` option to specify an alternate output directory.

You can specify the file path for source files that should be migrated. If using a compilation database, you can find your source code paths in the compilation database file.

The Folder Options sample shows an example of specifying a directory for migration and a specific output folder. For example:

Linux

```
dpct --in-root=foo --out-root=result/foo foo/main.cu foo/bar/util.cu
```

Windows

```
dpct --in-root=foo --out-root=result\foo foo\main.cu foo\bar\util.cu
```

For detailed instructions on how to use the Folder Options sample, refer to the sample README.

Use Make/CMake* to Migrate a Complete Project on Linux*

If your project uses Make or CMake, you can use a compilation database to provide compilation options, settings, macro definitions, and include paths to Intel® DPC++ Compatibility Tool. For example:

```
dpct --compilation-database=compile_commands.json --in-root=. --out-root=migration
```

Intel® DPC++ Compatibility Tool parses the compilation database and applies the necessary options when migrating the input sources. Refer to Generate a Compilation Database for detailed information about generating a compilation database.

The Needleman-Wunsch Sample shows an example of migrating a Make/CMake project, using a compilation database to provide project details to the tool.

For detailed instructions on how to use the Needleman-Wunsch sample, refer to the sample README.

Use Eclipse*

The Eclipse plugins are installed automatically when you specify an instance of Eclipse during the installation of the Intel® oneAPI Base Toolkit.

To access and use the samples within Eclipse:

1. Open Eclipse*
2. Select the **Intel®** menu, then **Browse Intel® oneAPI Samples**, and browse to the preferred sample
3. With the new project open, select **Migrate Project to DPC++**

   The migration creates a new project and warnings are displayed directly in the Eclipse* window
4. Verify the source for correctness and fix anything Intel® DPC++ Compatibility Tool could not migrate.

Use Microsoft Visual Studio*

You can use Intel® DPC++ Compatibility Tool within the Microsoft Visual Studio IDE to migrate your project to a SYCL project.

Step 1: Select an Existing Project and Start Migration

To select an existing project in Microsoft Visual Studio and start migration:

1. From the **Extensions** menu, select **Browse Intel® oneAPI Samples**
2. Select **Rodinia NW DPCT** from the **Intel® DPC++ Compatibility Tool** category.
3. Select the **Project** menu **>Migrate Project to DPC++**.
4. In the **Select Project to Migrate** dialog, click **Browse**.
5. Select the directory and the existing project file in the **Open** dialog.
6. Select the **Configuration** and **Platform** to migrate.
7. Click **OK**.

Step 2: Specify the Migration Configuration

After selecting an existing project, you will see the **Migration Configuration** dialog. These configurations have default values. You can specify the configurations, including the new project file name and directory, the `--in-root` argument, and the additional options for the migration.

The command line area is read-only and shows the command line that will be used to migrate the project.

The migration starts once you click **OK**.

Step 3: Verify the Source for Correctness

Verify the migration of the source code that uses variables declared using preprocessor directives. Inspect the migrated source code, address any DPCT warnings generated, and verify correctness of the new program.

For the most accurate and detailed instructions on addressing warnings, see the **Addressing Warnings in the Migrated Code** section of the sample README files.

Step 4: Check the Detailed Diagnostic Messages

To learn more about a diagnostic message, click the **Help** hyperlink in the **Actions** column, which is located in the table in the **Intel® DPC++ Compatibility Tool** view. Details about why the diagnostic message is generated and information on how to fix the issue are shown.

Specifying Options

You can set your preferences for Intel® DPC++ Compatibility Tool within Microsoft Visual Studio:

- To always show warnings after migration:

  1. Select **Tools** > **Options** > **Intel® DPC++ Compatibility Tool** > **General**.
  2. Select **True** (the default) for the **Always show warnings after migration** option.
  3. Click **OK**.
- To enable opening code side-by-side:

  1. Select **Tools** > **Options** > **Intel® DPC++ Compatibility Tool** > **General**.
  2. Select **True** (the default) for the **Enable opening code side-by-side** option.
  3. Click **OK**.
- To select a version of the tool:

  If you have multiple versions of Intel® DPC++ Compatibility Tool installed, you can select which version to use.

  1. Select **Tools** > **Options** > **Intel® DPC++ Compatibility Tool** > **Tools**.
  2. Use the drop-down menu from the step above to select the appropriate version of the tool.
  3. Select the default options (optional).
  4. Click **OK**.

Code Samples

Use the Intel® DPC++ Compatibility Tool code samples to get familiar with the migration process and tool features.

Each sample README provides detailed instructions for how to migrate the sample code.

Sample Project	Description
Vector Add	The Vector Add sample shows how to migrate a simple program from CUDA* to SYCL*. You can use this sample to verify that your development environment is set up correctly to use Intel® DPC++ Compatibility Tool.
Folder Options	The Folder Options sample shows how to migrate more complex projects and use tool options.

Sample Project	Description
Rodinia Needleman-Wunsch	The Rodinia Needleman-Wunsch sample demonstrates how to migrate a Make/CMake* project from CUDA to SYCL.

Explore the complete list of oneAPI code samples in the oneAPI Samples Catalog (GitHub). These samples were designed to help you develop, offload, and optimize multiarchitecture applications targeting CPUs, GPUs, and FPGAs.

To access the samples:

- Select a sample from the Intel® DPC++ Compatibility Tool category, using the `oneapi-cli` utility
- Download the sample from GitHub*.

The Intel® oneAPI Base Toolkit Get Started Guides more information on downloading samples. See also:

- Get Started Guide with the Intel® oneAPI Base Toolkit for Windows
- Get Started with the Intel® oneAPI Base Toolkit for Linux

Migration

This section contains information about migration workflow, planning a migration, and migration tool features.

Migration Workflow

The migration workflow guidelines describe the CUDA* to SYCL* code migration workflow, with detailed information about how to prepare your code for migration, how to use tool features to configure your migration, and how to build and optimize your migrated code.

Review Migration Workflow Guidelines.

Features to Support Migration

Intel® DPC++ Compatibility Tool provides multiple features to aid migration preparation and customization.

- Generate a Compilation Database
- Incremental Migration
- Migration Rules
- Migrate a CMake Build Script
- Debug Migrated Code: Runtime Behavior
- Analysis Mode

Query CUDA to SYCL API Mapping

You can query which SYCL* API is functionally compatible with a specified CUDA* API.

Learn how to Query CUDA* to SYCL* API Mapping.

Migration Workflow Guidelines

Overview

The CUDA* to SYCL* code migration workflow consists of the following high-level stages:

- Stage 1: Prepare for Migration. Prepare your project and configure the tool for a successful migration.
- Stage 2: Migrate Your Code. Review tool options and migrate your code with the tool.
- Stage 3: Review the Migrated Code. Review and manually convert any unmigrated code.
- Stage 4: Build the New SYCL Code Base. Build your project with the migrated code.
- Stage 5: Validate the New SYCL Application. Validate your new SYCL application to check for correct functionality after migration.

This document describes the steps in each stage with general recommendations and optional steps.

> **NOTE** CUDA* API migration support is broad but not complete. If you encounter CUDA APIs that were not migrated due to a lack of tool support, please report it to the Migrating to SYCL forum or priority support. Alternatively, submit an issue or contribute to the SYCLomatic project. This helps prioritize which CUDA APIs will be supported in future releases.

Prerequisites

- Install Intel® DPC++ Compatibility Tool. Intel® DPC++ Compatibility Tool is included in the Intel® oneAPI Base Toolkit (Base Kit). If you have not installed the Base Kit, follow the instructions in Install Intel® oneAPI Toolkits and Components.

  Intel® DPC++ Compatibility Tool is also available as a stand-alone download. Download the stand-alone tool.
- Set up the tool environment. Refer to Get Started with Intel® DPC++ Compatibility Tool for setup instructions.

Stage 1: Prepare for Migration

Before migrating your CUDA code to SYCL, prepare your CUDA source code for the migration process.

Prepare Your CUDA Project

Before migration, it is recommended to prepare your CUDA project to minimize errors during migration:

1. Make sure your CUDA source code has no syntax errors.
2. Make sure your CUDA source code is Clang compatible.

Fix Syntax Errors

If your original CUDA source code has syntax errors, it may result in unsuccessful migration.

Before you start migration, make sure that your original CUDA source code builds and runs correctly:

1. Compile your original source code using the compiler defined for your original CUDA project.
2. Run your compiled application and verify that it functions as expected.

When your code compiles with no build errors and you have verified that your application works as expected, your CUDA project is ready for migration.

Clang Compatibility

Intel® DPC++ Compatibility Tool uses the latest version of the Clang* parser to analyze your CUDA source code during migration. The Clang parser isn't always compatible with the NVIDIA* CUDA compiler driver (nvcc). The tool will provide errors about incompatibilities between nvcc and Clang during migration.

In some cases, additional manual edits to the CUDA source may be needed before migration. For example:

- The Clang parser may need namespace qualification in certain usage scenarios where nvcc does not require them.
- The Clang parser may need additional forward class declarations where nvcc does not require them.

  Space within the triple brackets of kernel invocation is tolerated by nvcc but not Clang. For example, `cuda_kernel<< <num_blocks, threads_per_block>> >(args…)` is ok for nvcc, but the Clang parser requires the spaces to be removed.

If you run the migration tool on CUDA source code that has unresolved incompatibilities between nvcc and Clang parsers, you will get a mixture of errors in the migration results:

- Clang errors, which must be resolved in the CUDA source code
- DPCT warnings, which must be resolved in the migrated SYCL code

For detailed information about dialect differences between Clang and nvcc, refer to llvm.org's Compiling CUDA with clang page.

Run CodePin to Capture Application Signature

CodePin is a feature that helps reduce the effort of debugging inconsistencies in runtime behavior. CodePin generates reports from the CUDA and SYCL programs that, when compared, can help identify the source of divergent runtime behavior.

Enable the CodePin tool during the migration in order to capture the project signature.

This signature will be used later for validation after migration.

Enable CodePin with the `–enable-codepin` option.

For detailed information about debugging using the CodePin tool, refer to Debug Migrated Code Runtime Behavior.

Configure the Tool

CUDA header files used by your project must be accessible to the tool. If you have not already done so, configure the tool and ensure header files are available.

Refer to Get Started with the Intel® DPC++ Compatibility Tool for installation and setup information.

Record Compilation Commands

Use `intercept-build` to Generate a Compilation Database to capture the detailed build options for your project. The migration tool uses build information from the database (such as header file paths, include paths, macro definitions, compiler options, and the original compiler) to guide the migration of your CUDA code.

If your development environment prevents you from using intercept-build, use the alternate method described in Generate a Compilation Database with Other Build Systems.

NOTE If you need to re-run your migration after the original migration and the CUDA build script has changed, you need to either

1. re-run intercept-build to get an updated compilation database to use in your migration or
2. manually update the compilation database to capture the changes from the updated CUDA build script.

Set Up Revision Control

After migration, the recommendation is to maintain and develop your migrated application in SYCL to avoid vendor lock-in, though you may choose to continue your application development in CUDA. Continuing to develop in CUDA will result in the need to migrate from CUDA to SYCL again.

Revision control allows comparison between versions of migrated code, which can help you decide what previous manual changes to the SYCL code you want to merge into the newly migrated code.

Make sure to have revision control for your original CUDA source before the first migration. After the first migration, be sure to place the migrated SYCL code, with all subsequent manual SYCL changes, under revision control as well.

Run Analysis Mode

You can use Analysis Mode to generate a report before migration that will indicate how much of your code will be migrated, how much will be partially migrated, and an estimate of the manual effort needed to complete migration after you have run the tool. This can be helpful to estimate the work required for your migration.

Stage 2: Migrate Your Code

Plan Your Migration

Before executing your migration, review the available tool features and options that can be used to plan your specific migration.

Migration Rules

The tool uses a default set of migration rules for all migrations. If default rules do not give the migration results you need, you can define custom rules for your migration. This is helpful in multiple scenarios, for example:

- After migration, you discover multiple instances of similar or identical CUDA source code that were not migrated, and you know how the CUDA source code should be migrated to SYCL. In this case, you can define a custom rule and re-run the migration for better results. This is useful for Incremental Migration or scenarios where you may run multiple migrations over time.
- You know before migration that some code patterns in your original CUDA source will not be accurately migrated to SYCL using the built-in rules. In this case, you can define a custom migration rule to handle specific patterns in our CUDA source during migration.

For detailed information about defining custom rules, refer to Migration Rules.

For working examples of custom rules, refer to the optional predefined rules located in the `extensions/opt_rules` folder on the installation path of the tool.

Incremental Migration

Intel® DPC++ Compatibility Tool provides incremental migration, which automatically merges the results from multiple migrations into a single migrated project.

Incremental migration can be used to

- migrate a CUDA* project incrementally, for example 10 files at a time
- migrate new CUDA files into an already migrated project
- migrate multiple code paths

Incremental migration is enabled by default. Disable incremental migration using the `--no-incremental-migration` option.

For detailed information and examples of incremental migration, refer to Incremental Migration.

Command-Line Options

Intel® DPC++ Compatibility Tool provides many command-line options to direct your migration. Command-line options provide control to

- Configure your migration with basic and advanced options
- Control code generation
- Generate migration reports
- Migrate build scripts
- Control warnings
- Query API mapping

Refer to the Alphabetical Option List for a full list of all available command-line options.

Buffer vs USM Code Generation

Intel promotes both buffer and USM in the SYCL/oneAPI context. Some oneAPI libraries preferentially support buffer versus USM, so there may be some design consideration in configuring your migration. USM is used by default, but buffer may be a better fit for some projects.

The buffer model sets up a 1-3 dimensional array (buffer) and accesses its components via a C++ accessor class. This grants more control over the exact nature and size of the allocated memory, and how host and offload target compute units access it. However, the buffer model can also create extra class management overhead, which can require more manual intervention and may yield less performance.

USM (unified shared memory) is a newer model, beginning with SYCL2020. USM is a pointer-based memory management model using `malloc_device/malloc_shared/malloc_host` allocator functions, similar to how C++ code usually handles memory accesses when no GPU device offload is involved. Choosing the USM model can make it easier to add to existing code and migrate from CUDA code. Management of the USM memory space is however very much done by the SYCL runtime, reducing granularity of control for the developer.

- SYCL Buffer Specification
- SYCL USM Specification

For more information on USM versus Buffer modes, please see the following sections of the GPU Optimization Guide: * Unified Shared Memory Allocations * Buffer Accessor Modes

What to Expect in Migrated Code

When the tool migrates CUDA code to SYCL code, it inserts diagnostic messages as comments in the migrated code. The DPCT diagnostic messages are logged as comments in the migrated source files and output as warnings to the console during migration. These messages identify areas in the migrated code that may require your attention to make the code SYCL compliant or correct. This step is detailed in Stage 3: Review the Migrated Code.

The migrated code also uses DPCT helper functions to provide utility support for the generated SYCL code. The helper functions use the `dpct:: namespace`. Helper function source files are located at `<tool-installation-directory>/latest/include/dpct`. DPCT helper functions can be left in migrated code but should not be used in new SYCL code. Use standard SYCL and C++ when writing new code. For information about the DPCT namespace, refer to the DPCT Namespace Reference.

Run Migration

After reviewing the available migration tool functionality and options, run your migration.

You can run the tool from the command line or within the Microsoft Visual Studio or Eclipse IDEs.

If your project uses a Makefile or CMake file, use the corresponding option to automatically migrate the file to work with the migrated code:

- To migrate a Makefile, use the `--gen-build-scripts` option.

- To migrate a CMake file, use the `--migrate-build-script` or `--migrate-build-script-only` option. (Note that these options are experimental.)

For example:

```
c2s -p compile_commands.json --in-root ../../.. --gen-helper-function --gen-build-scripts
```

This example migrate command:

- uses the tool alias `c2s`. `dpct` can also be used.
- uses a compilation database, specified with the `-p` option
- specifies the source to be migrated with the `--in-root` option
- instructs the tool to generate helper function files with the `--gen-helper-function` option
- instructs the tool to migrate the Makefile using the `--gen-build-scripts` option

The following samples show migrations of CUDA code using the tool and targets Intel and NVIDIA* hardware:

- convolutionSeparable Sample
- Concurrent Kernels Sample

Stage 3: Review the Migrated Code

After running Intel® DPC++ Compatibility Tool, manual editing is usually required before the migrated SYCL code can be compiled. DPCT warnings are logged as comments in the migrated source files and output to the console during migration. These warnings identify the portions of code that require manual intervention. Review these comments and make the recommended changes to ensure the migrated code is consistent with the original logic.

For example, this original CUDA* code:

```
void foo() {
  float *f;
  cudaError_t err = cudaMalloc(&f, 4);
  printf("%s\n", cudaGetErrorString(err));
}
```

results in the following migrated SYCL code:

```
void foo() {
  float *f;
  int err = (f = (float *)sycl::malloc_device(4, dpct::get_default_queue()), 0);
  /*
  DPCT1009:1: SYCL uses exceptions to report errors and does not use the error
  codes. The original code was commented out and a warning string was inserted.
  You need to rewrite this code.
  */
  printf("%s\n",
         "cudaGetErrorString is not supported" /*cudaGetErrorString(err)*/);
}
```

Note the DPCT1009 warning inserted where additional review is needed.

For a detailed explanation of the comments, including suggestions to fix the issues, refer to the Diagnostics Reference.

At this stage, you may observe that the same DPCT warnings were generated repeatedly in your code or that the same manual edits were needed in multiple locations to fix a specific pattern in your original source code. Consider defining the manual edits needed to fix repeated DPCT warnings as a user-defined migration rule. This allows you to save your corrections and automatically apply them to a future migration of your CUDA source.

NOTE CUDA* API migration support is broad but not complete. If you encounter CUDA APIs that were not migrated due to a lack of tool support, please report it to the Migrating to SYCL forum or priority support. Alternatively, submit an issue or contribute to the SYCLomatic project. This helps prioritize which CUDA APIs will be supported in future releases.

Stage 4: Build the New SYCL Code Base

After you have completed any manual migration steps, build your converted code.

Install New SYCL Code Base Dependencies

Converted code makes use of oneAPI library APIs and Intel SYCL extensions. Before compiling, install the appropriate oneAPI libraries and a compiler that supports the Intel SYCL extensions.

If your CUDA source uses ...	... install this oneAPI library
cuBLAS, cuFFT, cuRAND, cuSolver, cuSparse	Intel® oneAPI Math Kernel Library (oneMKL)
Thrust, CUB	Intel® oneAPI DPC++ Library (oneDPL)
cuDNN	Intel® oneAPI Deep Neural Network Library (oneDNN)
NCCL	Intel® oneAPI Collective Communications Library (oneCCL)

The following compilers support Intel SYCL extensions:

- Intel® oneAPI DPC++/C++ Compiler
- oneAPI DPC++ Compiler

Most libraries and the Intel® oneAPI DPC++/C++ Compiler are included in the Intel® oneAPI Base Toolkit (Base Kit). Libraries and the compiler are also available as stand-alone downloads.

Compile for Intel CPU and GPU

If your program targets Intel GPUs, install the latest Intel GPU drivers before compiling.

- Linux GPU Drivers
- Windows GPU Drivers

Use your updated Makefile or CMake file to build your program, or compile it manually at the command line using a compiler that supports the Intel SYCL extensions. Make sure that all linker and compilation commands use the `-fsycl` compiler option with the C++ driver. For example:

```
icpx -fsycl migrated-file.cpp
```

For detailed information about compiling with the Intel® oneAPI DPC++/C++ Compiler, refer to the Intel® oneAPI DPC++/C++ Compiler Developer Guide and Reference.

Compile for AMD* or NVIDIA* GPU

If your program targets AMD* or NVIDIA GPUs, install the appropriate Codeplay* plugin for the target GPU before compiling. Instructions for installing the AMD and NVIDIA GPU plugins, as well as how to compile for those targets, can be found in the Codeplay plugin documentation:

- Install the oneAPI for AMD GPUs plugin from Codeplay.
- Install the oneAPI for NVIDIA GPUs plugin from Codeplay.

Stage 5: Validate the New SYCL Application

After you have built your converted code, validate your new SYCL application to check for correct functionality after migration.

Use a Debugger to Validate Migrated Code

After you have successfully compiled your new SYCL application, run the app in debug mode using a debugger such as Intel Distribution for GDB to verify that your application runs as expected after migration.

Learn more about Debugging with Intel Distribution for GDB.

Use CodePIN to Validate Migrated Code

If the CodePin feature has been enabled during the migration time, project signature will be logged during the execution time.

The signature contains the data value of each execution checkpoint, which can be verified manually or with an auto-analysis tool.

For detailed information about debugging using the CodePin tool, refer to Debug Migrated Code Runtime Behavior.

Optimize Your Code

Optimize your migrated code for Intel GPUs using Intel® tools such as Intel® VTune™ Profiler and Intel® Advisor. These tools help identify areas of code to improve for optimizing your application performance.

Additional hardware- or library-specific optimization information is available:

- For detailed information about optimizing your code for Intel GPUs, refer to the oneAPI GPU Optimization Guide.
- For detailed information about optimizing your code for AMD GPUs, refer to the Codeplay AMD GPU Performance Guide.
- For detailed information about optimizing your code for NVIDIA GPUS, refer to the Codeplay NVIDIA GPU Performance Guide.

Find More

Content	Description
Intel® oneAPI DPC++/C++ Compiler Developer Guide and Reference	Developer guide and reference for the Intel® oneAPI DPC++/C++ Compiler.
SYCL 2020 Specification	The SYCL 2020 Specification PDF.
Intel® oneAPI DPC++/C++ Compiler	Intel branded C++ compiler built from the open-source oneAPI DPC++ Compiler, with additional Intel hardware optimization.
oneAPI DPC++ Compiler	Open-source Intel LLVM-based compiler project that implements compiler and runtime support for the SYCL* language.

Content	Description
Basic migration samples	Sample CUDA projects with instructions on migrating to SYCL using the tool.
Guided migration samples	Guided migration of two sample NVIDIA CUDA projects: • convolutionSeparable Sample • Concurrent Kernels Sample
Jupyter notebook samples	A Jupyter* Notebook that guides you through the migration of a simple example and four step-by-step sample migrations from CUDA to SYCL.
CUDA* to SYCL* Catalog	Catalog of CUDA projects that have been migrated to SYCL.
Migrating to SYCL forum	Forum to get assistance when migrating your CUDA code to SYCL.
Intel® oneAPI Math Kernel Library Link Line Advisor	Intel® oneAPI Math Kernel Library tool to help determine how to include oneMKL libraries for your specific use case.
Tutorial: Debugging with Intel® Distribution for GDB*	This tutorial describes the basic scenarios of debugging applications using Intel® Distribution for GDB*.
Intel® VTune™ Profiler Tutorials	Tutorials demonstrating an end-to-end workflow using Intel® VTune™ Profiler that you can ultimately apply to your own applications.

Generate a Compilation Database

A compilation database is a JSON file that contains an array of command objects, each of which usually contains three parts:

- `directory`: The working directory of the compilation. All paths specified in the command or file fields must be either absolute or relative to this directory.
- `command`: The compile command.
- `file`: The translation unit's main source file that is processed in the compilation step.

The compilation database lists detailed build options for each translation unit's main source file, which can be consumed by Intel® DPC++ Compatibility Tool to guide the code project migration.

For example:

```
[
  {
    "directory": "/path/to/project",
    "command": "/usr/bin/clang++ ...  -Iinclude_path -D_FOO_=VALUE -c ./foo/foo.cpp",
    "file": "./foo/foo.cpp"
  },
  ...
  {
    "directory": "/path/to/project",
    "command": "/usr/bin/clang++ ... -Iinclude_path -D_BAR_=VALUE -c ./foo/bar.cpp",
```

```
    "file": "./foo/bar.cpp"
  }
]
```

A compilation database can be generated by running the `intercept-build` script, which is provided as part of Intel® DPC++ Compatibility Tool. `intercept-build` supports the capture of compilation command lines for files with the following extensions: `.c`, `.C`, `.cc`, `.CC`, `.cp`, `.cpp`, `.cxx`, `.c++`, `.C++`, `.txx`, and `.cu`.

Generate a Compilation Database for Makefile-Based Projects

The source code in Makefile-based projects can be compiled and linked into an executable binary or library by running build commands specified in the project Makefile.

For Makefile-based projects, you can generate the compilation database by running `intercept-build make`.

Use the folder-options-dpct sample to show the generation of compilation database.

1. Change to the `folder-options-dpct` sample directory.
2. Run the intercept-build tool to generate the compilation database:

```
intercept-build make
```

   The compilation datebase `compile_commands.json` is created in the sample directory.

Generate a Compilation Database for CMake-Based Projects

CMake uses `CMakeLists.txt` files to describe project configuration, source files, and dependencies. When using CMake, make sure that all software that `CMakeLists.txt` files depend on has been installed, so that a functional Makefile can be generated by CMake.

There are two options to generate compilation database for CMake-based projects:

1. Use `intercept-build`
2. Use CMake option `CMAKE_EXPORT_COMPILE_COMMANDS`

Option One: Use intercept-build to Generate the Compilation Database

The following steps show how to generate the compilation database using `intercept-build`:

1. Install the required software dependencies for the project.
2. In a command window, go to the project folder.
3. In the project folder create and navigate to a new build directory:

```
mkdir build && cd build
```

4. Run CMake to generate a Makefile:

```
cmake ../
```

5. Run `intercept-build` to generate the compilation database:

```
intercept-build make
```

Option Two: Use CMake Option to Generate the Compilation Database

With CMake 3.5 version or higher, you can use the CMake option `CMAKE_EXPORT_COMPILE_COMMANDS` to generate the compilation database. For example, the following command generates the compilation database `compile_commands.json` in the build directory:

```
cmake  -DCMAKE_EXPORT_COMPILE_COMMANDS=ON ../
```

NOTE CMake 3.10 or higher is recommended as it provides better support for CUDA source code.

Generate a Compilation Database for Setuptools-Based Projects

Setuptools is a package development library in Python for facilitating packaging, distribution and installation of Python projects. For projects with setuptools as their build system, the steps to generate a compilation database are as follows:

1. Generate a build log for the setuptools project by using the `-v` or `--verbose` flag with the Python build command.

```
python3 setup.py <build-option> -v > build.log 2>&1
```

2. Use `intercept-build --parse-build-log` option to generate the compilation database.

```
intercept-build --parse-build-log build.log
```

Generate a Compilation Database with Other Build Systems

For projects using other build systems, `intercept-build` can be used to generate a compilation database from the build log of the project.

When generating a compilation database from the build log of a project, make sure to build your project using the appropriate build options to generate a verbose build log.

For example:

Build System	Option to Generate Verbose Build Log
Make	`VERBOSE=1`
CMake/Ninja	`-DCMAKE_VERBOSE_MAKEFILE=ON`
Bazel	`--subcommands`

You can use the folder-options-dpct sample to show the generation of the compilation database based on the build log.

1. Change to the `folder-options-dpct` sample directory.
2. Run `make` with options to collect a verbose build log:

```
make VERBOSE=1 -B > ./build_log.txt

 The content of ``build_log.txt`` should look like the following:

 .. code-block:: none
:linenos:

nvcc -c  -I./foo -I./foo/bar foo/main.cu -o foo/main.o
nvcc -c  -I./foo -I./foo/bar foo/bar/util.cu -o foo/bar/util.o
nvcc  ./foo/main.o ./foo/bar/util.o -o foo-bar
```

3. Use `intercept-build` to generate the compilation database with the `--parse-build-log` option to specify the build log to use:

```
intercept-build --parse-build-log=build_log.txt --work-directory=./
```

   The compilation database `compile_commands.json` is generated in the current directory.

For more information about intercept-build, run `intercept-build --help`.

Incremental Migration

Intel® DPC++ Compatibility Tool provides incremental migration, which automatically merges the results from multiple migrations into a single migrated project.

Incremental migration can be used to

- migrate a CUDA* project incrementally, for example 10 files at a time
- migrate new CUDA files into an already migrated project
- migrate multiple code paths

Incremental migration is enabled by default. Disable incremental migration using the `--no-incremental-migration` option.

Example 1: Migrate a File with Conditional Compilation Code

This example shows incremental migration for a file `sample1.cu` that contains conditional compilation code. Content of `sample1.cu`:

```
#ifndef MACRO_A
... code path 1 ...
#elif MACRO_A == 1
... code path 2 ...
#else
... code path 3 ...
#endif
```

Use the following steps to incrementally migrate `sample1.cu`.

1. Generate `sample1.dp.cpp`, which contains migrated code for code path 1. From the same working directory as the file, run:

```
dpct sample1.cu --out-root=out
```

2. Generate `sample1.dp.cpp`, which contains migrated code for code path 1 and code path 2:

```
dpct sample1.cu --out-root=out --extra-arg=”-DMACRO_A=1”
```

3. Generate `sample1.dp.cpp`, which contains migrated code for code path 1, code path 2, and code path 3:

```
dpct sample1.cu --out-root=out --extra-arg=”-DMACRO_A=2”
```

The result contains migrated code for each code path.

Example 2: Migrate a Header File Used by Multiple Source Files

This example shows the use of incremental migration for a header file that is included in several source files, each with a different macro definition.

Content of header file `sample_inc.h`:

```
#ifdef MACRO_A
... code path 1...
#else
... code path 2...
#endif
```

Content of source file `sample2.cu`:

```
#define MACRO_A
#include "sample_inc.h"
#undef MACRO_A
```

Content of source file `sample3.cu`:

```
#include "sample_inc.h"
```

Use the following steps to incrementally migrate the files.

1. Generate `sample2.dp.cpp` and `sample_inc.h`, which contains migrated code for code path 1.

   From the same working directory as the file, run:

```
dpct sample2.cu --out-root=out
```

2. Generate `sample3.dp.cpp` and `sample_inc.h`, which contains migrated code for code path 1 and code path 2:

```
dpct sample3.cu --out-root=out
```

The result contains migrated code for each code path.

Limitations

Incremental migration will not be triggered in the following conditions:

1. Intel® DPC++ Compatibility Tool option `--no-incremental-migration` is specified.
2. Different versions of Intel® DPC++ Compatibility Tool are used across multiple migration invocations.
3. Different options of Intel® DPC++ Compatibility Tool are used across multiple migration invocations. If Intel® DPC++ Compatibility Tool detects that a previous migration used a different option-set, Intel® DPC++ Compatibility Tool will stop migration and exit.

The following options direct Intel® DPC++ Compatibility Tool to generate different migrated code and may break incremental migration. Use the same values for these options across migration invocations to keep incremental migration working.

- `--always-use-async-handler`
- `--assume-nd-range-dim`
- `--comments`
- `--custom-helper-name`
- `--enable-ctad`
- `--keep-original-code`
- `--no-cl-namespace-inline`
- `--no-dpcpp-extensions`

- `--no-dry-pattern`
- `--optimize-migration`
- `-p`
- `--process-all`
- `--sycl-named-lambda`
- `--use-experimental-features`
- `--use-explicit-namespace`
- `--usm-level`
- `--vcxprojfile`

Migration Rules

Intel® DPC++ Compatibility Tool uses migration rules to migrate CUDA* code to SYCL* code. There are three categories of migration rules used by the tool:

- **Default migration rules.** A set of built-in migration rules used by the tool for all migrations.
- **Optional predefined migration rules.** A set of predefined migration rules that can optionally be used for migration. Available predefined migration rules are in the *extensions/opt_rules* folder on the installation path of the tool.
- **User-defined migration rules.** Custom migration rules defined by the user. User-defined migration rules extend the migration capability of Intel® DPC++ Compatibility Tool and can be used to target the migration of specific CUDA syntax to specific SYCL syntax.

Specify Migration Rule Files

To specify a predefined or user-defined migration rule file for use in migration, use the `–rule-file` command line option with your migration command.

The `–rule-file` option can be used multiple times with a single command to specify multiple migration rule files. For example:

```
dpct sample.cu --rule-file=rule_file1.YAML --rule-file=rule_file2.YAML
```

See the Command Line Options Reference for additional information.

Write User-defined Migration Rules

Migration rules are specified in YAML files. A single rule file may contain multiple migration rules. To define a rule, use the following <key>: <value> pairs:

Key	Value	Description
Rule	String value	Required. Specifies the unique name of the rule.
Priority	`Takeover` | `Default` | `Fallback`	Required. Specifies the priority of the rule: `Takeover` > `Default` > `Fallback`. When there are rule conflicts, the rule with higher priority will take precedence.
Kind	`Macro` | `API` | `Header` | `Type` | `Class` | `Enum` | `DisableAPIMigration` |	Required. Specifies the rule type.

Key	Value	Description
	`PatternRewriter` | `CMakeRule` | `PythonRule`	
CmakeSyntax	String value	Required. Specify the CMake syntax name that will be migrated. Use the unique name for the CMake syntax.
PythonSyntax	String value	Optional. Specify the Python syntax name that will be migrated. Use the unique name for the Python syntax.
MatchMode	`Partial` | `Full` | `StrictFull`	Optional. Specify the match mode with full word match, strict full word match, or partial word match. If not specified, partial match mode will be used. The partial matching mode means the matched string can be surrounded by arbitrary characters including whitespace. The full matching mode means the matched string should not be surrounded by identifier characters (letters, numbers, and underscore). The strict matching mode means the matched string should only be surrounded by whitespace characters.
In	String value	Required. Specifies the target name in the input source code.
Out	String value	Required. Specifies the final format in the output source code.
Includes	List of header files	Required. Specifies the header files which should be included in the output source code. The value can be an empty list.
EnumName	String value	Specifies the name of an enum for an Enum rule type.
Fields	String value	Specifies the migration rule of fields in a Class rule type.
Methods	String value	Specifies the migration rule of methods in a Class rule type.
Prefix	String value	Specifies the prefix of a Header rule type. For example: `#ifdef ...`
Postfix	String value	Specifies the postfix of a Header rule type. For example: `#endif ...`
Subrules	String value	Specifies the subrules for the PatternRewriter rule type.
APIRestrictCondition	List of restrictions	Specifies the restrictions for the API.

For example, the following user-defined migration rule file demonstrates different rule types. The behavior of each rule is explained in the corresponding comment:

```
---                                  # [YAML syntax] Begin the document
- Rule: rule_forceinline             # Rule to migrate "__forceinline__" to "inline"
  Kind: Macro                        # Rule type
  Priority: Takeover                 # Rule priority
  In: __forceinline__                # Target macro name in the input source code
  Out: inline                        # Name of migrated macro in the output source code
  Includes: ["header1.h", "\"header2.h\""]  # List of header file names which the new macro
                                     # depends on
- Rule: rule_foo                     # Rule to migrate "foo(a,b)" to "int *new_ptr=bar(*b)"
  Kind: API
```

```
  Priority: Takeover
  In: foo                              # Target function name in the input source code
  Out: $type_name_of($2) *new_ptr = bar($deref($1))  # Format of the migrated result in the
                                                     # output source code
  Includes: ["<header3>"]
  APIRestrictCondition:
    ArgCount: 1                        # Only the overloading with 1 argument will be migrated
- Rule: rule_cmath                     # Rule to migrate "include<cmath>" to "#include<mymath>"
  Kind: Header
  Priority: Takeover
  In: cmath
  Out: mymath
  Prefix: "#ifdef USE_MYMATH\n"        # Add prefix before "#include<mymath>"
  Postfix: "#endif\n"                  # Add postfix after "#include<mymath>"
  Includes: [""]
- Rule: rule_classA                    # Rule to migrate "classA" to "classB"
  Kind: Class
  Priority: Takeover
  In: classA
  Out: classB
  Includes: []
  Fields:                              # Specify the migration rule of fields of classA
    - In: fieldA                       # Migrate classA.fieldA to getter and setter
      OutGetter: get_a                 # Migrate value reference of classA.fieldA to
                                       # classB.get_a()
      OutSetter: set_a                 # Migrate value assignment of classA.fieldA to
                                       # classB.set_a()
    - In: fieldC
      Out: fieldD                      # Migrate classA.fieldC to classB.fieldD
  Methods:
    - In: methodA
      Out: a.methodB($2)               # Migrate classA.methodA(x,y) to a.methodB(y)
- Rule: rule_Fruit                     # Rule to migrate "Fruit:apple" to "Fruit:pineapple"
  Kind: Enum
  Priority: Takeover
  EnumName: Fruit
  In: apple
  Out: pineapple
  Includes: ["fruit.h"]
- Rule: type_rule                      # Migrate "OldType" to "NewType"
  Kind: Type
  Priority: Takeover
  In: OldType
  Out: NewType
  Includes: []
- Rule: disable_rule                   # Disable the migration of an API
  Kind: DisableAPIMigration
  Priority: Takeover
  In: foo                              # Disable the migration of foo
  Out: ""
  Includes: []
- Rule: post_migration_rewriter_rule   # Post-migration pattern rewrite rule which uses nested
                                       # string pattern search and replace to find and update
                                       # strings in the migrated code
  Kind: PatternRewriter
  Priority: Takeover
  In: my_max(${args});                 # Match pattern "my_max(...);" and save the arbitrary
                                       # arbitrary string between "my_max(" and ");" as ${args}.
```

```
                                    # "args" can be a user-defined name which will be
                                    # referenced by "Out" and "Subrules"
  Out: my_min(${args});             # Replace the pattern string to "my_min(${args});"
  Includes: []
  Subrules:
    args:                           # Specify the subrule to apply to ${args}, where "args"
                                    # is the user-defined name which is defined in "In".
      In: a                         # Match pattern "a" in ${args}
      Out: b                        # Replace the pattern string to "b" in ${args}
...                                 # [YAML syntax] End the document
```

Grammar for Out Key in a User-defined API Migration Rule

To describe the value format for the `Out` key in a migration rule of `Kind: API`, use the following Backus-Naur form grammar:

```
OutValue::= Token | Token OutValue       # OutValue is the value for the "out" key
Token::= AnyString | Keyword             # AnyString is a string provided by the user
Keyword::= ArgIndex
   | $queue                              # Represents the queue string
   | $context                            # Represents the context string
   | $device                             # Represents the device string
   | $deref(ArgIndex)                    # The dereferenced value of the argument
   | $type_name_of(ArgIndex)             # The type name of the argument
   | $deref_type(ArgIndex)               # The dereferenced type name of the argument
   | $addr_of(ArgIndex)                  # The address of the argument
ArgIndex::= $Int                         # Int should be a greater than zero integer
```

The following scenario describes how the tool makes use of a user-defined migration rule that uses this grammar to migrate code.

Consider the following user-defined API migration rule:

```
- Rule: rule_foo
  Kind: API
  Priority: Takeover
  In: foo
  Out: $type_name_of($2) new_ptr = bar($deref($1), $3)
  Includes: ["<header3>"]
```

If the input source code contains a function call that matches the rule, the tool parses the value of the `In` and `Out` keys and builds a keyword mapping between the input and output source code. For example, with input source code:

```
int *ptr, *ptr2;
foo(ptr, ptr2, 30);
```

The tool creates the following mapping:

Keyword	Input Source Code Match	Migration Result
$1	ptr	ptr

Keyword	Input Source Code Match	Migration Result
`$2`	`ptr2`	`ptr2`
`$3`	`30`	`30`
`$type_name_of($2)`	N/A	`int*`
`$deref($1)`	N/A	`*ptr`

Using this mapping, the tool migrates the input source code into the following output source code:

```
int *ptr, *ptr2;
int * new_ptr = bar(*ptr, 30);
```

Migrate a CMake Build Script

Intel® DPC++ Compatibility Tool supports migration of CMake build scripts by migrating a CUDA* specific CMake syntax to the SYCL* version. The migration capability can be extended and customized with a user-defined migration rule, which can be loaded with option `--rule-file`. The tool provides two command line options to trigger the migration of the CMake build script:

Option	Description
`--migrate-build-script=CMake`	EXPERIMENTAL. Migrate the CMake file(s).
`--migrate-build-script-only`	EXPERIMENTAL. Only migrate the build script(s). Default: off.

Examples

Example 1: Migrate the CMake build script of a CUDA project:

```
dpct -in-root=./ -out-root=out  -migrate-build-script-only
```

Example 2: Migrate the CMake build script of a CUDA project using a user-defined migration rule defined in `rule_file.yaml`:

```
dpct -in-root=./ -out-root=out  -migrate-build-script-only  --rule-file=rule_file.yaml
```

Example 3: Migrate the CMake build script of a CUDA project together with source code migration:

```
dpct -p=./ -in-root=./ -out-root=out  --migrate-build-script=CMake
```

NOTE View a demo example of CMake-based apps in Velocity-Bench.

Customize Migration of CMake Build Script

To customize migration of a CMake build script, you can use the predefined built-in migration rules, or write a custom user-defined migration rule.

Use Predefined Migration Rules for CMake Build Script

The tool provides a set of predefined migration rules in the `extensions/cmake_rules`

folder on the install path. These predefined rules may be helpful in migrating your CMake build script.

Use the `-rule-file` option to specify the use of a predefined migration rule.

Write a Custom User-defined Migration Rule

You can customize the migration of your CMake file by defining a user-defined migration rule. Detailed information about writing your own user-defined migration rule, is provided in the Write User-defined Migration Rules section.

The following example demonstrates a user-defined CMake build script migration rule. The functionality of each field of the rule is detailed in the comment column. The example rule targets the CMake macro `cuda_compile_ptx` for migration, migrating `cuda_compile_ptx(obj_file kernel.cu)` to `dpct_helper_compile_sycl_code (obj_file kernel.dp.cpp)`.

> **NOTE** `dpct_helper_compile_sycl_code()` is provided by the tool. It is located in `dpct.cmake` in the `cmake` folder on the install path, during the CMake script migration, `dpct.cmake` will be copied to the root folder of the migrated project and as a part of the migrated project.

```
- Rule: rule_cuda_compile_ptx                # Specify the unique name of the
                                             rule
  Kind: CMakeRule                            # The kind of the rule
  Priority: Fallback                         # The priority of the rule
                                             (Takeover, Default, Fallback)
  CmakeSyntax: cuda_compile_ptx              # The CMake syntax target to
                                             migrate
  In: cuda_compile_ptx(${device} ${value}) # Specify target pattern of
                                             cuda_compile_ptx will be
                                             migrated
  Out: dpct_helper_compile_sycl_code(${device} ${value}) # Specify the
                                                           content of to migrate the content
specified in "In" file
     Subrules:                               # Subrules to further process
                                             the field(s) of pattern
                                             specified in "In" field
       value:                                # Process the value file of
                                             pattern specified in "In" file
         MatchMode: Full                     # Specify the match mode of the
                                             sub rule Full or Partial, default is Partial
         In: ${arg}.cu                       # Replace the file name in *.cu
                                             pattern to *.cp.cpp
         Out: ${arg}.dp.cpp
```

For additional information on defining user-defined migration rules, refer to Migration Rules.

To write a user defined migration rule for CMake syntax:

1. Identify the CMake CUDA specific syntax that needs to be migrated. It is important to understand the usage case(s) of the CMake CUDA specific syntax.
2. Figure out the functionally equivalent SYCL-side mapping for CUDA CMake syntax, and test the mapping. If SYCL side functionally equivalent mapping is not available, consider adding a helper function, such as the helper function available in the `dpct.cmake` file on the tool install path.
3. Abstract the CMake syntax mapping between CUDA and SYCL into a pattern-based description in a yaml-based rule.

4. Test the functionality of the rule with lit cases by running the following command to make sure the migration rule is working as expected. If not, you may need to refine the migration rule defined in step 2.

```
dpct -in-root=./  -out-root out -rule-file=new_rule.yaml test.cmake
```

5. Use the new rule(s) to migrate your project.

Debug Migrated Code: Runtime Behavior

NOTE The CodePin feature is experimental and will be finalized in a future release.

In some cases, the migrated SYCL* program may have runtime behavior that differs from the original CUDA* program. Reasons for this inconsistency may include:

- Difference in arithmetic precision between hardware
- Semantic difference between the CUDA and SYCL APIs
- Errors introduced during the automatic migration

CodePin is feature of Intel® DPC++ Compatibility Tool that helps reduce the effort of debugging such inconsistencies in runtime behavior. When CodePin is enabled, Intel® DPC++ Compatibility Tool will migrate the CUDA program to SYCL, but will also generate an instrumented version of the CUDA program.

The instrumented code will dump the data of related variables, before and after selected API or kernel calls, into a report. Compare the reports generated from the CUDA and SYCL programs to help identify the source of divergent runtime behavior.

Enable CodePin

Enable CodePin with the `-enable-codepin` option. The instrumented program will be put in the folder `dpct_output_codepin_cuda` and `dpct_output_codepin_sycl`.

Example

The following example CUDA code has an issue in the cudaMemcpy() before the vectorAdd kernel call: the size to be copied is hard coded as `vectorSize * 12` instead of `vectorSize * sizeof(int3)`, which causes incorrect behavior of the migrated SYCL program. This is because `int3` will be migrated to `sycl::int3` and the size of `sycl::int3` is 16 bytes, not 12 bytes.

```
//example.cu
#include <iostream>
__global__ void vectorAdd(int3 *a, int3 *result) {
    int tid = blockIdx.x * blockDim.x + threadIdx.x;
    result[tid].x = a[tid].x + 1;
    result[tid].y = a[tid].y + 1;
    result[tid].z = a[tid].z + 1;
}

int main() {
    const int vectorSize = 4;
    int3 h_a[vectorSize], h_result[vectorSize];
    int3 *d_a, *d_result;
    for (int i = 0; i < vectorSize; ++i)
        h_a[i] = make_int3(1, 2, 3);

    cudaMalloc((void **)&d_a, vectorSize * sizeof(int3));
    cudaMalloc((void **)&d_result, vectorSize * sizeof(int3));
```

```
    // Copy host vectors to device
    // !! Using 12 instead of "sizeof(int3)"
    cudaMemcpy(d_a, h_a, vectorSize * 12, cudaMemcpyHostToDevice);

    // Launch the CUDA kernel
    vectorAdd<<<1, 4>>>(d_a, d_result);

    // Copy result from device to host
    cudaMemcpy(h_result, d_result, vectorSize * sizeof(int3),
        cudaMemcpyDeviceToHost);

    // Print the result
    for (int i = 0; i < vectorSize; ++i) {
        std::cout << "Result[" << i << "]: (" << h_result[i].x << ", "
                << h_result[i].y << ", " << h_result[i].z << ")\n";
    }
}

/*
Execution Result:
Result[0]: (2, 3, 4)
Result[1]: (2, 3, 4)
Result[2]: (2, 3, 4)
Result[3]: (2, 3, 4)
*/
```

To debug the issue, the migrate the CUDA program with CodePin enabled:

```
dpct example.cu --enable-codepin
```

After migration, there will be two files: `dpct_output_codepin_sycl/example.dp.cpp` and `dpct_output_codepin_cuda/example.cu`.

```
workspace
├── example.cu
├── dpct_output_codepin_sycl
│   ├── example.dp.cpp
│   ├── generated_schema.hpp
│   └── MainSourceFiles.yaml
├── dpct_output_codepin_cuda
│   ├── example.cu
│   └── generated_schema.hpp
```

`dpct_output_codepin_sycl/example.dp.cpp` is the migrated and instrumented SYCL program:

```
//dpct_output_codepin_sycl/example.dp.cpp
#include <dpct/dpct.hpp>
#include <sycl/sycl.hpp>

#include "generated_schema.hpp"
#include <dpct/codepin/codepin.hpp>
#include <iostream>

void vectorAdd(sycl::int3 *a, sycl::int3 *result,
            const sycl::nd_item<3> &item_ct1) {
    int tid = item_ct1.get_group(2) * item_ct1.get_local_range(2) +
                item_ct1.get_local_id(2);
    result[tid].x() = a[tid].x() + 1;
    result[tid].y() = a[tid].y() + 1;
    result[tid].z() = a[tid].z() + 1;
```

```
}

int main() {
    sycl::device dev_ct1;
    sycl::queue q_ct1(dev_ct1,
                      sycl::property_list{sycl::property::queue::in_order()});
    const int vectorSize = 4;
    sycl::int3 h_a[vectorSize], h_result[vectorSize];
    sycl::int3 *d_a, *d_result;
    for (int i = 0; i < vectorSize; ++i)
        h_a[i] = sycl::int3(1, 2, 3);

    d_a = sycl::malloc_device<sycl::int3>(vectorSize, q_ct1);
    dpct::experimental::get_ptr_size_map()[*((void **)&d_a)] =
        vectorSize * sizeof(sycl::int3);

    d_result = sycl::malloc_device<sycl::int3>(vectorSize, q_ct1);
    dpct::experimental::get_ptr_size_map()[*((void **)&d_result)] =
        vectorSize * sizeof(sycl::int3);

    // Copy host vectors to device
    q_ct1.memcpy(d_a, h_a, vectorSize * 12);

    // Launch the CUDA kernel
    dpctexp::codepin::gen_prolog_API_CP(
        "vectorAdd:example.cu:24:9",
        &q_ct1, "d_a", d_a, "d_result", d_result);
    q_ct1.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 4), sycl::range<3>(1, 1, 4)),
        [=](sycl::nd_item<3> item_ct1) { vectorAdd(d_a, d_result, item_ct1); });

    // Copy result from device to host
    dpctexp::codepin::gen_epilog_API_CP(
        "vectorAdd:example.cu:24:9",
        &q_ct1, "d_a", d_a, "d_result", d_result);

    q_ct1.memcpy(h_result, d_result, vectorSize * sizeof(sycl::int3)).wait();

    // Print the result
    for (int i = 0; i < vectorSize; ++i) {
        std::cout << "Result[" << i << "]: (" << h_result[i].x() << ", "
                << h_result[i].y() << ", " << h_result[i].z() << ")\n";
    }
}

/*
Execution Result:
Result[0]: (2, 3, 4)
Result[1]: (2, 3, 4)
Result[2]: (2, 3, 4)
Result[3]: (1, 1, 1) <--- incorrect result
*/
```

`dpct_output_codepin_cuda/example.cu` is the instrumented CUDA program:

```
//dpct_output_codepin_cuda/example.cu
#include "generated_schema.hpp"
#include <dpct/codepin/codepin.hpp>
#include <iostream>
```

```
__global__ void vectorAdd(int3 *a, int3 *result) {
    int tid = blockIdx.x * blockDim.x + threadIdx.x;
    result[tid].x = a[tid].x + 1;
    result[tid].y = a[tid].y + 1;
    result[tid].z = a[tid].z + 1;
}

int main() {
    const int vectorSize = 4;
    int3 h_a[vectorSize], h_result[vectorSize];
    int3 *d_a, *d_result;
    for (int i = 0; i < vectorSize; ++i)
        h_a[i] = make_int3(1, 2, 3);

    cudaMalloc((void **)&d_a, vectorSize * sizeof(int3));
    dpct::experimental::get_ptr_size_map()[*((void **)&d_a)] =
        vectorSize * sizeof(int3);
    cudaMalloc((void **)&d_result, vectorSize * sizeof(int3));
    dpct::experimental::get_ptr_size_map()[*((void **)&d_result)] =
        vectorSize * sizeof(int3);

    // Copy host vectors to device
    cudaMemcpy(d_a, h_a, vectorSize * 12, cudaMemcpyHostToDevice);

    // Launch the CUDA kernel
    dpctexp::codepin::gen_prolog_API_CP(
        "vectorAdd:example.cu:24:9", 0,
        "d_a", d_a, "d_result", d_result);
    vectorAdd<<<1, 4>>>(d_a, d_result);

    // Copy result from device to host
    dpctexp::codepin::gen_epilog_API_CP(
        "vectorAdd:example.cu:24:9", 0,
        "d_a", d_a, "d_result", d_result);
    cudaMemcpy(h_result, d_result, vectorSize * sizeof(int3),
               cudaMemcpyDeviceToHost);

    // Print the result
    for (int i = 0; i < vectorSize; ++i) {
        std::cout << "Result[" << i << "]: (" << h_result[i].x << ", "
                << h_result[i].y << ", " << h_result[i].z << ")\n";
    }
}

/*
Execution Result:
Result[0]: (2, 3, 4)
Result[1]: (2, 3, 4)
Result[2]: (2, 3, 4)
Result[3]: (2, 3, 4)
*/
```

After building `dpct_output_codepin_sycl/example.dp.cpp` and `dpct_output_codepin_cuda/example.cu` and executing the binaries built out the following execution log files will be generated.

Report for the instrumented CUDA program

```
[
    {
        "ID": "example.cu:26:3:prolog",
        "Device Name": "GPU",
        "Device ID": "0",
        "Stream Address": "0xe4bb30",
        "Free Device Memory": "16374562816",
        "Total Device Memory": "16882663424",
        "Elapse Time(ms)": "0",
        "CheckPoint": {
            "d_a": {
                "Type": "Pointer",
                "Data": [
                    {
                        "Type": "int3",
                        "Data": [
                            {
                                "x": {
                                    "Type":
"int",
                                    "Data": [
                                        1
                                    ]
                                }
                            },
                            {
                                "y": {
                                    "Type":
"int",
                                    "Data": [
                                        2
                                    ]
                                }
                            },
  ...
```

Report for the instrumented migrated SYCL program

```
[
    {
        "ID": "example.cu:26:3:prolog",
        "Device Name": "GPU",
        "Device ID": "0",
        "Stream Address": "0x3fea40",
        "Free Device Memory": "0",
        "Total Device Memory": "31023112192",
        "Elapse Time(ms)": "0",
        "CheckPoint": {
            "d_a": {
                "Type": "Pointer",
                "Data": [
                    {
                        "Type": "sycl::int3",
                        "Data": [
                            {
                                "x": {
                                    "Type":
"int",
                                    "Data": [
                                        1
                                    ]
                                }
                            },
                            {
                                "y": {
                                    "Type":
"int",
                                    "Data": [
                                        2
                                    ]
                                }
                            },
    ...
```

The report helps identify where the runtime behavior of the CUDA and the SYCL programs start to diverge from one another.

Analyse the CodePin Result

CodePin Report

codepin-report.py (also can be triggered by dpct/c2s –codepin-report) is a functionality of the compatibility tool that consumes the execution log files from both CUDA and SYCL code and performs auto analysis. codepin-report.py can identify the inconsistent data value and report the stats data of the execution.

User can specify the comparison tolerance of floating points in the following format:

```
{
    "bf16_abs_tol": 9.77e-04,
    "fp16_abs_tol": 9.77e-04,
    "float_abs_tol": 1.19e-04,
```

```
    "double_abs_tol": 2.22e-04,
    "rel_tol": 1e-3
}
```

The first 4 items “bf16_abs_tol”, “bf16_abs_tol”, “bf16_abs_tol” and “bf16_abs_tol” are the absolute tolerance range of the corresponding type. The last item, “rel_tol”, is the relative tolerance represented by a ratio value.

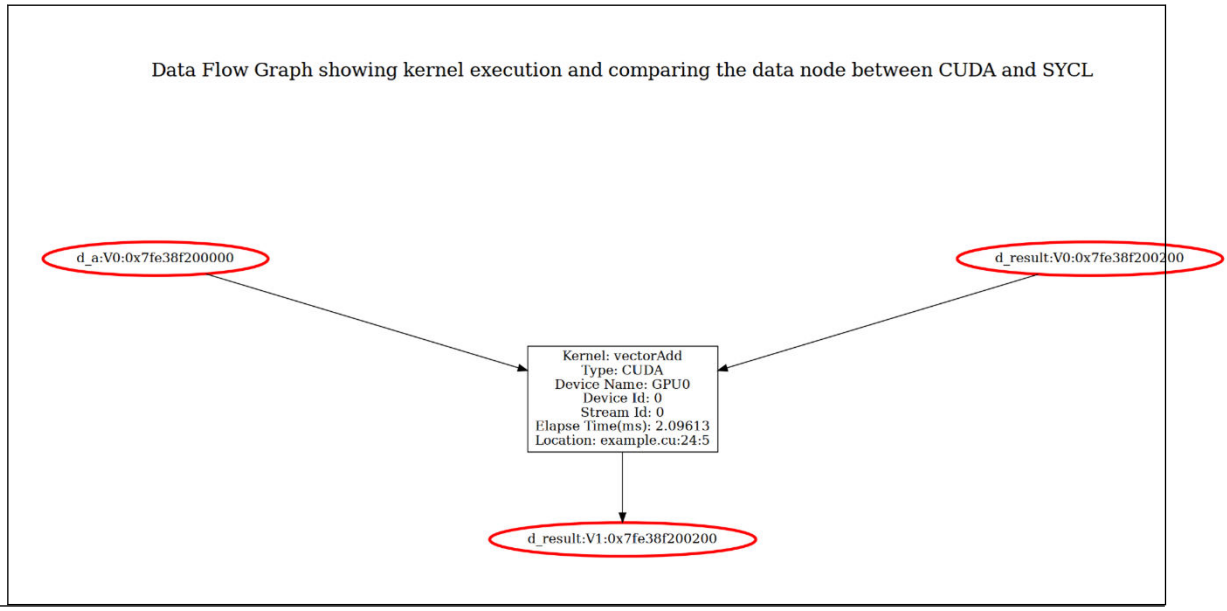
codepin-report.py consumes the execution log files generated from both CUDA and SYCL code with the following command line. `codepin-report.py [-h] --instrumented-cuda-log <file path> --instrumented-sycl-log <file path> [--floating-point-comparison-epsilon <file path>]`

Following is an example of the analysis report.

```
CodePin Summary
Total API count, 2
Consistent API count, 0
Most Time-consuming Kernel(CUDA), vectorAdd:example.cu:24:5:epilog, time:16.8069
Most Time-consuming Kernel(SYCL), vectorAdd:example.cu:24:5:prolog, time:18.3240
Peak Device Memory Used(CUDA), 445644800
Peak Device Memory Used(SYCL), 540689534976
CUDA Meta Data ID, SYCL Meta Data ID, Type, Detail
vectorAdd:example.cu:24:5:epilog,vectorAdd:example.cu:24:5:epilog,Data value,The location of
failed ID Errors occurred during comparison: d_a->"Data"->[3]->"Data"->[0]->"x"->"Data"->[0] and
[ERROR: DATA VALUE MISMATCH] the CUDA value 1 differs from the SYCL value 26518016.; d_result-
>"Data"->[3]->"Data"->[0]->"x"->"Data"->[0] and [ERROR: DATA VALUE MISMATCH] the CUDA value 2
differs from the SYCL value 26518017.
vectorAdd:example.cu:24:5:prolog,vectorAdd:example.cu:24:5:prolog,Data value,[WARNING: METADATA
MISMATCH] The pair of prolog data vectorAdd:example.cu:24:5:prolog are mismatched, and the
corresponding pair of epilog data matches. This mismatch may be caused by the initialized memory
or argument used in the API vectorAdd.
```

Data Flow Graph

codepin-report.py can generate a data flow graph for kernels with option `--generate-data-flow-graph`. The data flow graph presents visualizations of kernel execution and compares results between CUDA and SYCL, highlighting the execution mismatch between CUDA and SYCL code. In the data flow graph, each kernel execution and its input and output arguments are grouped into a layer, presenting a run status of the kernel execution. The value of input and output arguments are tagged with version information in the form of “V<num>”. For example, the initial version is tagged as V0, and once the value of the argument is updated, the version number will be increased. For a specific kernel execution, if there’s a mismatch between CUDA and SYCL results, the mismatched argument node will be colored red.



The above picture shows the data flow graph of the vectorAdd example, which is constructed by a title and execution layer. The execution layer presents a kernel execution and its inputs and outputs. The kernel node shows that kernel `vectorAdd` is executed on the stream of the device named GPU0, and also shows the kernel's execution time and source location. All input arguments (`d_a` node and top `d_result` node) are tagged with V0, indicating initial values. The output argument (bottom `d_result` node) is tagged with V1 because `d_result` is both input and output arguments, and its value changes in the kernel.

The nodes `d_a:V0`, `d_result:V0`, and `d_result:V1` are colored red, indicating a value mismatch between the CUDA and SYCL runs. In this case, the result value mismatch is caused by the mismatch of the input argument values, and the mismatch between input argument values may be caused by the different behavior of memory initialization between CUDA and SYCL, as the report states.

This data flow graph target provides a clear view of the execution process, making it easy to identify discrepancies and track variable changes across executions.

Query CUDA* to SYCL* API Mapping

Use the `--query-api-mapping` option to discover which SYCL* API is functionally compatible with a specified CUDA* API. The API query functionality can assist users on manual code migration and help users understand the SYCL API.

The following example queries the functionally compatible SYCL API for the `cudaMalloc` function:

```
$ dpct --query-api-mapping=cudaMalloc

  CUDA API:
    cudaMalloc(pDev /*void ***/, s /*size_t*/);
  Is migrated to:
    *pDev = (void *)sycl::malloc_device(s, dpct::get_in_order_queue());
```

On Linux, you can use the `tab` key to auto-complete the CUDA API name specified for the `--query-api-mapping` option. For example :

1. Specify the partial CUDA function name “cudaMa” to the `--query-api-mapping` option:

```
dpct --query-api-mapping=cudaMa
```

2. Press `tab`.

   The CUDA API name will auto-complete to `cudaMalloc`:

```
dpct --query-api-mapping=cudaMalloc
```

Intel® DPC++ Compatibility Tool also supports fuzzy matching of CUDA API names. For example:

```
$ dpct --query-api-mapping=cudamalloc

  CUDA API:
    cudaMalloc(pDev /*void ***/, s /*size_t*/);
  Is migrated to:
    *pDev = (void *)sycl::malloc_device(s, dpct::get_in_order_queue());
```

Analysis Mode

Analysis Mode generates a summary report that shows:

- Lines of code that are migrated with confidence
- Lines of code that are migrated imperfectly due to the language gap between CUDA* and SYCL*
- An estimation of how much human effort will be required to complete the code migration

All migration issues are classified into one of three levels of manual effort:

- **Low Effort**: Indicates an issue that the developer can solve in minutes.
- **Medium Effort**: Indicates an issue that may take hours to solve.
- **High Effort**: Indicates that there is not yet equivalent functionality provided in SYCL or SYCL ecosystem libraries for the CUDA API or the CUDA library API. In these cases, the developer will need to manually rewrite the functionality.

An example analysis report can be seen as follows:

```
reduction.cpp:
  + 26 lines of code (100%) will be automatically migrated.
    - 22 APIs/Types - No manual effort.
    -  4 APIs/Types - Low manual effort for checking and code fixing.
    -  0 APIs/Types - Medium manual effort for code fixing.
  +  0 lines of code (  0%) will not be automatically migrated.
    -  0 APIs/Types - High manual effort for code fixing.
reduction_kernel.cu:
  + 77 lines of code ( 89%) will be automatically migrated.
    -  7 APIs/Types - No manual effort.
    - 63 APIs/Types - Low manual effort for checking and code fixing.
    -  7 APIs/Types - Medium manual effort for code fixing.
  +  9 lines of code ( 10%) will not be automatically migrated.
    -  9 APIs/Types - High manual effort for code fixing.
Total Project:
  +103 lines of code ( 91%) will be automatically migrated.
```

```
    - 29 APIs/Types - No manual effort.
    - 67 APIs/Types - Low manual effort for checking and code fixing.
    -  7 APIs/Types - Medium manual effort for code fixing.
+   9 lines of code (  8%) will not be automatically migrated.
    -  9 APIs/Types - High manual effort for code fixing.
```

Intel® DPC++ Compatibility Tool provides a command line option to enable this feature:

```
$ dpct --analysis-mode sample.cu
```

Intel® DPC++ Compatibility Tool supports saving the report to a file:

```
$ dpct --analysis-mode --analysis-report-output-file=report.txt sample.cu
```

Reference

This section contains Intel® DPC++ Compatibility Tool reference information and information about mapping CUDA* concepts and APIs to SYCL*.

- Command Line Options Reference
- Diagnostics Reference
- DPCT Namespace Reference
- CUDA* and SYCL* Programming Model Comparison
- CUDA* to SYCL* Term Mapping Quick Reference

Command Line Options Reference

The tool supports many command line options you can use in your migration. This section provides:

- An alphabetical list of command line options
- Functional groups of options:
  - Basic Migration Options
  - Advanced Migration Options
  - Code Generation Options
  - Report Generation Options
  - Build Script Options
  - Query API Mapping, Tool Warning, and Help Options
- A list of intercept-build options

Auto-Complete Command Line Options

The tool supports auto-completion of command line options. You can auto-complete the option name and option value (for enumeration options only) by using <tab>.

Enable this feature with the following command:

```
source /path/to/oneapi/package/install/folder/<toolkit-version>/oneapi-vars.sh
```

> **NOTE** Auto-completion of command line options is only supported for Bash on Linux*.
> The auto-complete feature will not work if <tab> is used after <space>.

The following examples show how to use the auto-complete feature:

- Show all tool options:

```
dpct --<tab><tab>
```

- Auto-complete the command to `dpct --version`

```
dpct --ver<tab>
```

- Show all available values of the `--use-experimental-features` option:

```
dpct --use-experimental-features=<tab><tab>
```

- Auto-complete the command to `dpct --use-experimental-features=logical-group,matrix`:

```
dpct --use-experimental-features=logical-group,mat<tab>
```

Basic Migration Options

The following table lists command line options for basic migration.

Option	Description
`--analysis-scope-path=<dir>`	The directory path for the analysis scope of the source tree that needs to be migrated. Default: the value of `--in-root`.
`--compilation-database=<dir>`	The directory path for the compilation database (compile_commands.json) for the files to be migrated. Paths and build options described in the project file are used to guide the migration. When no path is specified, a search for compile_commands.json is attempted through all parent directories of the first input source file. Same as -p.
`--cuda-include-path=<dir>`	The directory path of the CUDA* header files.

Option	Description
`--extra-arg=<string>`	Specify additional arguments to append to the migration command line. The option accepts a comma-separated list of extra arguments and may be used multiple times in a command. For example: `dpct xx.cu --extra-arg="-I /path, -I /path2" --extra-arg="-I /path/to/header"` Use the `dpct -- -help` command to find options that can be passed as extra arguments for the migration command.
`--in-root=<dir>`	The directory path for the root of the source tree that needs to be migrated. Only files under this root are migrated. Default: • The current directory, if the input source files are not provided. • The directory of the first input source file, if the input source files are provided. Details: • Any source within the directory specified by `--in-root` (at any nesting level) may be migrated. • Any header file within the directory specified by `--in-root` (at any nesting level) that is included by the source or header file which is being migrated, is also migrated. • Files from outside the `--in-root` directory will not be migrated even if they are included by any of the program source files.
`--in-root-exclude=<dir|file>`	Exclude the specified directory or file from processing.
`--no-incremental-migration`	Tell the tool to not perform an incremental migration. Default: `off` (incremental migration happens).
`--out-root=<dir>`	The directory path for root of generated files. A directory is created if it does not exist. Default: `dpct_output`. The relative paths for the generated files are maintained. By default, file extensions are changed as follows: • `*.cu` → `*.dp.cpp` • `*.cpp` → `*.cpp.dp.cpp` • `*.cc` → `*.cc.dp.cpp` • `*.cxx` → `*.cxx.dp.cpp` • `*.C` → `*.C.dp.cpp` • `*.cuh` → `*.dp.hpp` • `*.h *.hpp *.hxx` → extensions are kept the same To limit file extension changes to `.cu` and `.cuh` files only, use the `--change-cuda-files-extension-only` option.
`-p`	Alias for `--compilation-database`.
`--process-all`	Migrate or copy all files, except hidden, from the `--in-root` directory to the `--out-root` directory. The `--in-root` option should be explicitly specified. Default: `off`. Details: If `--process-all` and `--in-root` options are specified, but no input files are provided, the tool migrates or copies all files, except hidden, from the `--in-root` directory to the output directory.

Option	Description
	• If there is a compilation database: • Files from the compilation database are migrated with the options specified in the compilation database • Files with the `.cu` extension that are not listed in the compilation database are migrated as standalone • Remaining files are copied to the `-out-root` directory • If there is no compilation database: • Files with the `.cu` extension are migrated as standalone • Remaining files are copied to the `-out-root` directory `--process-all` is ignored if input files are provided on the command line.
`--stop-on-parse-err`	Stop migration and generation of reports if parsing errors happened. Default: `off`.
`--vcxprojfile=<file>`	The directory path for the C++ project file `vcxproj` of the Visual Studio project to migrate. Paths and build options described in the project file are used to guide the migration.
`--intercept-build`	Intercept build tool to generate a compilation database.
`--sycl-file-extension=<value>`	Specify the extension of migrated source file(s). The values are: • `=dp-cpp`: Use extension '.dp.cpp' and '.dp.hpp' (default). • `=sycl-cpp`: Use extension '.sycl.cpp' and '.sycl.hpp'. • `=cpp`: Use extension '.cpp' and '.hpp'.

Advanced Migration Options

The following tables list advanced options to control migrated source, generate reports, and query API mapping.

Options to Format Migrated Code

Option	Description
`--format-range=<value>`	Set the range of formatting. The values are: • `=all`: Format all code. • `=migrated`: Only format the migrated code (default). • `=none`: Do not format any code.
`--format-style=<value>`	Set the formatting style. The values are: • `=custom`: Use the coding style defined in the `.clang-format` file (default). • `=google`: Use the Google* coding style. • `=llvm`: Use the LLVM coding style.

Option	Description

	Example for the .clang-format file content:
	`BasedOnStyle:`
	`LLVM IndentWidth: 4`
	`TabWidth: 4`
	`UseTab: ForIndentation`

Options to Specify Additional Migration Rules

Option	Description
`--rule-file=<file>`	Specify the rule file path that contains rules used for migration.

Options to Control SYCL Extensions and Experimental Features

Option	Description
`--no-dpcpp-extensions=<value>`	A comma-separated list of extensions not to be used in migrated code. By default, these extensions are used in migrated code. The values are: • `=bfloat16`: Disable the SYCL extensions for bfloat16. See more details. • `=device_info`: Disable the Intel extensions for device information, if supported by the compiler and the backend. See more details. • `=enqueued_barriers`: Disable the enqueued barriers extension. See more details. • `=peer_access`: Disable the peer access extension. See more details. • `=assert`: Disable the assert extension. See more details. • `=queue_empty`: Disable the queue empty extension. See more details. • `=all`: Disable all extensions listed in this option.
`--use-dpcpp-extensions=<value>`	A comma-separated list of extensions to be used in migrated code. By default, these extensions are not used in migrated code. • `=c_cxx_standard_library`: Use std functions from the libdevice library (provided by Intel® oneAPI DPC++/C++ Compiler) and C/C++ Standard Library to migrate functions which have no mapping in the SYCL standard. If this value is used together with `intel_device_math`, the `intel_device_math` functions take precedence. See more details. • `=intel_device_math`: Use `sycl::ext::intel::math` functions from the libdevice library (provided by Intel® oneAPI DPC++/C++ Compiler) to migrate functions which have no mapping in the SYCL standard. See more details.

Option	Description
	• `all`: Enable all DPC++ extensions listed in this option.
`--use-experimental-features=<value>`	A comma-separated list of experimental features to be used in migrated code. By default, experimental features will not be used in migrated code. The values are: • `=bfloat16_math_functions`: Experimental extension that allows use of bfloat16 math functions. See more details. • `=bindless_images`: Experimental extension that allows use of bindless images APIs. See more details. • `=dpl-experimental-api`: Experimental extension that allows use of experimental oneDPL APIs. See more details. • `=free-function-queries`: Experimental extension that allows getting `id`, `item`, `nd_item`, `group`, and `sub_group` instances globally. See more details. • `=local-memory-kernel-scope-allocation`: Experimental extension that allows allocation of local memory objects at the kernel functor scope. See more details. • `=logical-group`: Experimental helper function used to logically group work-items. See more details in `dpct::experimental::logical_group` in header file `util.hpp`. • `=masked-sub-group-operation`: Experimental helper function used to execute sub-group operation with mask. See more details in `dpct::experimental::select_from_sub_group`, `dpct::experimental::shift_sub_group_left`, `dpct::experimental::shift_sub_group_right` and `dpct::experimental::shift_sub_group_right` in header file `util.hpp`. • `=matrix`: Experimental extension that allows use of matrix extension like class `joint_matrix`. See more details. • `=nd_range_barrier`: DEPRECATED. Experimental helper function used to help cross-group synchronization during migration. Please use the following option instead: `--use-experimental-features=root-group` • `=root-group`: Experimental extension that allows use of root group class and relative API. • `=graph`: Experimental extension that allows use of SYCL Graph APIs. • `=occupancy-calculation`: Experimental helper function used to calculate occupancy. See more details in `dpct::experimental::calculate_max_active_wg_per_xecore` and `dpct::experimental::calculate_max_potential_wg` in header file `util.hpp`. • `=user-defined-reductions`: Experimental extension that allows user-defined reductions. See more details. • `=non-uniform-groups`: Experimental extension that allows use of non-uniform groups. See more details. • `=device_global`: Experimental extension that allows device scoped memory allocations into SYCL that can be accessed within a kernel using syntax similar to C++ global variables. See more details. • `=all`: Enable all experimental extensions listed in this option.

Code Generation Options

The following table lists command line options to customize how your code is migrated.

Option	Description
`--always-use-async-handler`	Use async exception handler when creating new `sycl::queue` with `dpct::create_queue` in addition to default `dpct::get_default_queue`. Default: `off`.
`--assume-nd-range-dim=<value>`	Provide a hint to the tool on the dimensionality of nd_range to use in generated code. The values are: • `=1`: Generate kernel code assuming 1D `nd_range` where possible, and 3D in other cases. • `=3`: Generate kernel code assuming 3D `nd_range` (default).
`--change-cuda-files-extension-only`	Limit extension change to `.cu` and `.cuh` files only. Default: `off`.
`--check-unicode-security`	Enable detection and warnings about Unicode constructs that can be exploited by using bi-directional formatting codes and homoglyphs in identifiers. Default: `off`.
`--comments`	Insert comments explaining the generated code. Default: `off`.
`--enable-ctad`	Use a C++17 class template argument deduction (CTAD) in your generated code.
`--enable-profiling`	Enable SYCL* queue profiling in helper functions. Default: `auto` (when set to `auto`, the `enable-profiling` option will only be used if the tool deduces that profiling is required during migration).
`--gen-helper-function`	Generate helper function files in the `--out-root` directory. Default: `off`.
`--helper-function-preference=<value>`	The preference of helper function usage in migration. Value: • `=no-queue-device`: Call SYCL API to get queue and device instead of calling helper function.
`--keep-original-code`	Keep the original code in the comments of generated SYCL files. Default: `off`.
`--no-dry-pattern`	Do not use a Don't Repeat Yourself (DRY) pattern when functions from the `dpct` namespace are inserted. Default: `off`.
`--optimize-migration`	Generate SYCL code applying more aggressive assumptions that potentially may alter the semantics of your program. Default: `off`.
`--sycl-named-lambda`	Generate kernels with the kernel name. Default: `off`.
`--use-explicit-namespace=<value>`	Define the namespaces to use explicitly in generated code. The value is a comma-separated list. Default: `dpct/syclcompat, sycl`. Possible values are:

Option	Description
	• =dpct: Generate code with dpct:: namespace. • =syclcompat: Generate code with syclcompat:: namespace. • =none: Generate code without any namespaces. Cannot be used with other values. • =sycl: Generate code with sycl:: namespace. Cannot be used with cl or sycl-math values. • =sycl-math: Generate code with sycl:: namespace, applied only for SYCL math functions. Cannot be used with cl or sycl values.
--usm-level=<value>	Set the Unified Shared Memory (USM) level to use in source code generation: • =none: Uses helper functions from Intel® DPC++ Compatibility Tool header files for memory management migration. • =restricted: Uses USM API for memory management migration. (default).
--enable-codepin	EXPERIMENTAL: Generate instrumented CUDA and SYCL code for debug and verification purposes in the directory <dir>_codepin_cuda and <dir>_codepin_sycl, where <dir> is specified by --out-root option.

Report Generation Options

The following table lists command line options to control report generation during migration.

Option	Description
--report-file-prefix=<prefix>	Specify the prefix for the migration report file names. The full file name will have a suffix derived from the report-type, and an extension derived from the report-format. For example: <prefix>.apis.csv or <prefix>.stats.log. If this option is not specified, the report goes to stdout. The report files are created in the directory, specified by -out-root.
--report-format=<value>	Specify the format of the reports: • =csv: Output will be lines of comma-separated values. The report file name extension will .csv (default). • =formatted: Output will be formatted for easier readability. Report file name extension will be log.
--report-only	Generate migration reports only. No SYCL code will be generated. Default: off.
--report-type=<value>	Specify the type of migration report. Values are: • =all: All of the migration reports.

Option	Description
	• `=apis`: Information about API signatures that need migration and the number of times they were encountered. The report file name has the `.apis` suffix added. • `=stats`: High level migration statistics: Lines Of Code (LOC) that are migrated to SYCL, LOC migrated to SYCL with helper functions, LOC not needing migration, LOC needing migration but are not migrated. The report file name has the `.stats` suffix added (default).
`--analysis-mode`	Only generate a report for porting effort. Default: `off`.
`--analysis-mode-output-file`	Specify the file where the analysis mode report is saved. Default: Output to `stdout`.

Build Script Options

The following table lists command line options to migrate build scripts.

Option	Description
`--build-script-file=<file>`	Specify the name of generated makefile for migrated file(s). Default name: `Makefile.dpct`.
`--gen-build-script`	Generate makefile for migrated file(s) in `-out-root` directory. Default: `off`.
`--migrate-build-script=<value>`	EXPERIMENTAL: Migrate build script(s). • `=CMake`: Migrate the CMake file(s).
`--migrate-build-script-only`	EXPERIMENTAL: Only migrate the build script(s). Default: `off`.

Query API Mapping, Tool Warning, and Help Options

The following tables list command line options to query API mapping support, manage warnings generated by the tool, and to display tool information.

Query API Mapping Options

Option	Description
`-query-api-mapping=<api>`	Query functionally-compatible SYCL API to migrate CUDA API.

Tool Warning Options

Option	Description
`--output-file=<file>`	Redirect the `stdout`/`stderr` output to `<file>` in the output directory specified by the `--out-root` option.
`--output-verbosity=<value>`	Set the output verbosity level: • `=detailed`: 'normal' and messages about which file is being processed. • `=diagnostics`: 'detailed' and information about the detected conflicts and crashes (default). • `=normal`: 'silent' and warnings, errors, and notes from Intel® DPC++ Compatibility Tool. • `=silent`: Only messages from clang.
`--suppress-warnings=<value>`	A comma-separated list of migration warnings to suppress. Valid warning IDs range from 1000 to 1132. Hyphen-separated ranges are also allowed. For example: `-suppress-warnings=1000-1010,1011`.
`--suppress-warnings-all`	Suppress all migration warnings. Default: `off`.

Help and Info Options

Option	Description
`--help`	Display available options.
`--helper-function-dir`	Print the installation directory for helper function header files.
`--version`	Show the version of the tool.

intercept-build Options

The **intercept-build** tool is available for Linux* only. It is available as part of Intel® DPC++ Compatibility Tool installation.

The following table lists **intercept-build** tool command line options.

Option	Description
–append	Extend existing compilation database with new entries. Duplicate entries are detected and not present in the final output. The output is not continuously updated; it's done when the build command finished. Default: disabled.
–cdb <file>	The JSON compilation database. Default name: **compile_commands.json**.
–intercept-build	Invoke the **intercept-build** tool to generate a compilation database.
–linker-entry	Generate linker entry in compilation database if the **–linker-entry** option is present. Default: enabled.
–no-linker-entry	Do not generate linker entry in compilation database if the **–no-linker-entry** option is present. Default: disabled.
–parse-build-log <file>	Specify the file path of the build log.
–verbose, **-v**	Enable verbose output from **intercept-build**. A second, third, and fourth flag increases verbosity.
–work-directory <path>	Specify the working directory of the command that generates the build log specified by option **-parse-build-log**. Default: the directory of build log file specified by option **-parse-build-log**.

Deprecated Options

Occasionally, tool options are marked as "deprecated." Deprecated options are still supported in the current release, but they may be removed in future releases.

There are currently no deprecated options.

Alphabetical Option List

The following table lists Intel® DPC++ Compatibility Tool command line options in alphabetical order.

Option	Description
`--always-use-async-handler`	Use async exception handler when creating new `sycl::queue` with `dpct::create_queue` in addition to default `dpct::get_default_queue`. Default: `off`.

Option	Description
`--analysis-mode`	Only generate a report for porting effort. Default: `off`.
`--analysis-mode-output-file`	Specify the file where the analysis mode report is saved. Default: Output to `stdout`.
`--analysis-scope-path=<dir>`	The directory path for the analysis scope of the source tree that needs to be migrated. Default: the value of `--in-root`.
`--assume-nd-range-dim=<value>`	Provide a hint to the tool on the dimensionality of nd_range to use in generated code. The values are: • `=1`: Generate kernel code assuming 1D `nd_range` where possible, and 3D in other cases. • `=3`: Generate kernel code assuming 3D `nd_range` (default).
`--build-script-file=<file>`	Specify the name of generated makefile for migrated file(s). Default name: `Makefile.dpct`.
`--change-cuda-files-extension-only`	Limit extension change to `.cu` and `.cuh` files only. Default: `off`.
`--check-unicode-security`	Enable detection and warnings about Unicode constructs that can be exploited by using bi-directional formatting codes and homoglyphs in identifiers. Default: `off`.
`--comments`	Insert comments explaining the generated code. Default: `off`.
`--compilation-database=<dir>`	The directory path for the compilation database (compile_commands.json) for the files to be migrated. Paths and build options described in the project file are used to guide the migration. When no path is specified, a search for compile_commands.json is attempted through all parent directories of the first input source file. Same as -p.
`--cuda-include-path=<dir>`	The directory path of the CUDA* header files.
`--enable-codepin`	EXPERIMENTAL: Generate instrumented CUDA and SYCL code for debug and verification purposes in the directory `<dir>_codepin_cuda` and `<dir>_codepin_sycl`, where `<dir>` is specified by `--out-root` option.
`--enable-ctad`	Use a C++17 class template argument deduction (CTAD) in your generated code.
`--enable-profiling`	Enable SYCL* queue profiling in helper functions. Default: `auto` (when set to `auto`, the `enable-profiling` option will only be used if the tool deduces that profiling is required during migration).
`--extra-arg=<string>`	Specify additional arguments to append to the migration command line. The option accepts a comma-separated list of extra arguments and may be used multiple times in a command. For example: `dpct xx.cu --extra-arg="-I /path, -I /path2" --extra-arg="-I /path/to/header"` Use the `dpct -- -help` command to find options that can be passed as extra arguments for the migration command.
`--format-range=<value>`	Set the range of formatting.

Option	Description
	The values are: • `=all`: Format all code. • `=migrated`: Only format the migrated code (default). • `=none`: Do not format any code.
`--format-style=<value>`	Set the formatting style. The values are: • `=custom`: Use the coding style defined in the `.clang-format` file (default). • `=google`: Use the Google* coding style. • `=llvm`: Use the LLVM coding style. Example for the .clang-format file content: `BasedOnStyle: LLVM IndentWidth: 4 TabWidth: 4 UseTab: ForIndentation`
`--gen-build-script`	Generate makefile for migrated file(s) in `-out-root` directory. Default: `off`.
`--gen-helper-function`	Generate helper function files in the `--out-root` directory. Default: `off`.
`--help`	Display available options.
`--helper-function-dir`	Print the installation directory for helper function header files.
`--helper-function-preference=<value>`	The preference of helper function usage in migration. Value: • `=no-queue-device`: Call SYCL API to get queue and device instead of calling helper function.
`--intercept-build`	Intercept build tool to generate a compilation database.
`--in-root=<dir>`	The directory path for the root of the source tree that needs to be migrated. Only files under this root are migrated. Default: • The current directory, if the input source files are not provided. • The directory of the first input source file, if the input source files are provided. Details: • Any source within the directory specified by `--in-root` (at any nesting level) may be migrated. • Any header file within the directory specified by `--in-root` (at any nesting level) that is included by the source or header file which is being migrated, is also migrated. • Files from outside the `--in-root` directory will not be migrated even if they are included by any of the program source files.
`--in-root-exclude=<dir|file>`	Exclude the specified directory or file from processing.
`--keep-original-code`	Keep the original code in the comments of generated SYCL files. Default: `off`.

Option	Description
`--migrate-build-script=<value>`	EXPERIMENTAL: Migrate build script(s). • `=CMake`: Migrate the CMake file(s).
`--migrate-build-script-only`	EXPERIMENTAL: Only migrate the build script(s). Default: `off`.
`--no-dpcpp-extensions=<value>`	A comma-separated list of extensions not to be used in migrated code. By default, these extensions are used in migrated code. The values are: • `=bfloat16`: Disable the SYCL extensions for bfloat16. See more details. • `=device_info`: Disable the Intel extensions for device information, if supported by the compiler and the backend. See more details. • `=enqueued_barriers`: Disable the enqueued barriers extension. See more details. • `=peer_access`: Disable the peer access extension. See more details. • `=assert`: Disable the assert extension. See more details. • `=queue_empty`: Disable the queue empty extension. See more details. • `=all`: Disable all extensions listed in this option.
`--no-dry-pattern`	Do not use a Don't Repeat Yourself (DRY) pattern when functions from the `dpct` namespace are inserted. Default: `off`.
`--no-incremental-migration`	Tell the tool to not perform an incremental migration. Default: `off` (incremental migration happens).
`--optimize-migration`	Generate SYCL code applying more aggressive assumptions that potentially may alter the semantics of your program. Default: `off`.
`--out-root=<dir>`	The directory path for root of generated files. A directory is created if it does not exist. Default: `dpct_output`. The relative paths for the generated files are maintained. By default, file extensions are changed as follows: • `*.cu → *.dp.cpp` • `*.cpp → *.cpp.dp.cpp` • `*.cc → *.cc.dp.cpp` • `*.cxx → *.cxx.dp.cpp` • `*.C → *.C.dp.cpp` • `*.cuh → *.dp.hpp` • `*.h *.hpp *.hxx` → extensions are kept the same To limit file extension changes to `.cu` and `.cuh` files only, use the `--change-cuda-files-extension-only` option.
`--output-file=<file>`	Redirect the `stdout`/`stderr` output to `<file>` in the output directory specified by the `--out-root` option.
`--output-verbosity=<value>`	Set the output verbosity level: • `=detailed`: 'normal' and messages about which file is being processed. • `=diagnostics`: 'detailed' and information about the detected conflicts and crashes (default). • `=normal`: 'silent' and warnings, errors, and notes from Intel® DPC++ Compatibility Tool.

Option	Description
	• `=silent`: Only messages from clang.
`-p`	Alias for `--compilation-database`.
`--process-all`	Migrate or copy all files, except hidden, from the `--in-root` directory to the `--out-root` directory. The `--in-root` option should be explicitly specified. Default: `off`. Details: If `--process-all` and `--in-root` options are specified, but no input files are provided, the tool migrates or copies all files, except hidden, from the `--in-root` directory to the output directory. • If there is a compilation database: • Files from the compilation database are migrated with the options specified in the compilation database • Files with the `.cu` extension that are not listed in the compilation database are migrated as standalone • Remaining files are copied to the `-out-root` directory • If there is no compilation database: • Files with the `.cu` extension are migrated as standalone • Remaining files are copied to the `-out-root` directory `--process-all` is ignored if input files are provided on the command line.
`-query-api-mapping=<api>`	Query functionally-compatible SYCL API to migrate CUDA API.
`--report-file-prefix=<prefix>`	Specify the prefix for the migration report file names. The full file name will have a suffix derived from the `report-type`, and an extension derived from the `report-format`. For example: `<prefix>.apis.csv` or `<prefix>.stats.log`. If this option is not specified, the report goes to `stdout`. The report files are created in the directory, specified by `-out-root`.
`--report-format=<value>`	Specify the format of the reports: • `=csv`: Output will be lines of comma-separated values. The report file name extension will `.csv` (default). • `=formatted`: Output will be formatted for easier readability. Report file name extension will be `log`.
`--report-only`	Generate migration reports only. No SYCL code will be generated. Default: `off`.
`--report-type=<value>`	Specify the type of migration report. Values are: • `=all`: All of the migration reports. • `=apis`: Information about API signatures that need migration and the number of times they were encountered. The report file name has the `.apis` suffix added. • `=stats`: High level migration statistics: Lines Of Code (LOC) that are migrated to SYCL, LOC migrated to SYCL with helper functions, LOC not needing migration, LOC needing migration but are not migrated. The report file name has the `.stats` suffix added (default).
`--rule-file=<file>`	Specify the rule file path that contains rules used for migration.

Option	Description
`--stop-on-parse-err`	Stop migration and generation of reports if parsing errors happened. Default: `off`.
`--suppress-warnings=<value>`	A comma-separated list of migration warnings to suppress. Valid warning IDs range from 1000 to 1132. Hyphen-separated ranges are also allowed. For example: `-suppress-warnings=1000-1010,1011`.
`--suppress-warnings-all`	Suppress all migration warnings. Default: `off`.
`--sycl-file-extension=<value>`	Specify the extension of migrated source file(s). The values are: • `=dp-cpp`: Use extension '.dp.cpp' and '.dp.hpp' (default). • `=sycl-cpp`: Use extension '.sycl.cpp' and '.sycl.hpp'. • `=cpp`: Use extension '.cpp' and '.hpp'.
`--sycl-named-lambda`	Generate kernels with the kernel name. Default: `off`.
`--use-dpcpp-extensions=<value>`	A comma-separated list of extensions to be used in migrated code. By default, these extensions are not used in migrated code. • `=c_cxx_standard_library`: Use std functions from the libdevice library (provided by Intel® oneAPI DPC++/C++ Compiler) and C/C++ Standard Library to migrate functions which have no mapping in the SYCL standard. If this value is used together with `intel_device_math`, the `intel_device_math` functions take precedence. See more details. • `=intel_device_math`: Use `sycl::ext::intel::math` functions from the libdevice library (provided by Intel® oneAPI DPC++/C++ Compiler) to migrate functions which have no mapping in the SYCL standard. See more details. • `all`: Enable all DPC++ extensions listed in this option.
`--use-experimental-features=<value>`	A comma-separated list of experimental features to be used in migrated code. By default, experimental features will not be used in migrated code. The values are: • `=bfloat16_math_functions`: Experimental extension that allows use of bfloat16 math functions. See more details. • `=bindless_images`: Experimental extension that allows use of bindless images APIs. See more details. • `=dpl-experimental-api`: Experimental extension that allows use of experimental oneDPL APIs. See more details. • `=free-function-queries`: Experimental extension that allows getting `id`, `item`, `nd_item`, `group`, and `sub_group` instances globally. See more details. • `=local-memory-kernel-scope-allocation`: Experimental extension that allows allocation of local memory objects at the kernel functor scope. See more details. • `=logical-group`: Experimental helper function used to logically group work-items. See more details in `dpct::experimental::logical_group` in header file `util.hpp`. • `=masked-sub-group-operation`: Experimental helper function used to execute sub-group operation with mask. See more details in `dpct::experimental::select_from_sub_group`, `dpct::experimental::shift_sub_group_left`, `dpct::experimental::shift_sub_group_right` and `dpct::experimental::shift_sub_group_right` in header file `util.hpp`.

Option	Description
	• `=matrix`: Experimental extension that allows use of matrix extension like class `joint_matrix`. See more details. • `=nd_range_barrier`: DEPRECATED. Experimental helper function used to help cross-group synchronization during migration. Please use the following option instead: `--use-experimental-features=root-group` • `=root-group`: Experimental extension that allows use of root group class and relative API. • `=graph`: Experimental extension that allows use of SYCL Graph APIs. • `=occupancy-calculation`: Experimental helper function used to calculate occupancy. See more details in `dpct::experimental::calculate_max_active_wg_per_xecore` and `dpct::experimental::calculate_max_potential_wg` in header file `util.hpp`. • `=user-defined-reductions`: Experimental extension that allows user-defined reductions. See more details. • `=non-uniform-groups`: Experimental extension that allows use of non-uniform groups. See more details. • `=device_global`: Experimental extension that allows device scoped memory allocations into SYCL that can be accessed within a kernel using syntax similar to C++ global variables. See more details. • `=all`: Enable all experimental extensions listed in this option.
`--use-explicit-namespace=<value>`	Define the namespaces to use explicitly in generated code. The value is a comma-separated list. Default: `dpct/syclcompat, sycl`. Possible values are: • `=dpct`: Generate code with `dpct::` namespace. • `=syclcompat`: Generate code with `syclcompat::` namespace. • `=none`: Generate code without any namespaces. Cannot be used with other values. • `=sycl`: Generate code with `sycl::` namespace. Cannot be used with `cl` or `sycl-math` values. • `=sycl-math`: Generate code with `sycl::` namespace, applied only for SYCL math functions. Cannot be used with `cl` or `sycl` values.
`--use-syclcompat`	Use SYCLcompat header-only library (`syclcompat::` namespace) to assist the migration of input source code. Default: `off`.
`--usm-level=<value>`	Set the Unified Shared Memory (USM) level to use in source code generation: • `=none`: Uses helper functions from Intel® DPC++ Compatibility Tool header files for memory management migration. • `=restricted`: Uses USM API for memory management migration. (default).
`--vcxprojfile=<file>`	The directory path for the C++ project file `vcxproj` of the Visual Studio project to migrate. Paths and build options described in the project file are used to guide the migration.
`--version`	Show the version of the tool.

Diagnostics Reference

This topic gives you all of the diagnostic reference numbers, messages, detailed help, and suggestion to fix your errors.

ID	Message
DPCT1000	Error handling `if-stmt` was detected but could not be rewritten.
DPCT1001	The statement could not be removed.
DPCT1002	Special case error handling `if-stmt` was detected. You may need to rewrite this code.
DPCT1003	Migrated API does not return error code. `(*, 0)` is inserted. You may need to rewrite this code.
DPCT1004	REMOVED. Compatible SYCL code could not be generated.
DPCT1005	The SYCL device version is different from CUDA Compute Compatibility. You may need to rewrite this code.
DPCT1006	SYCL does not provide a standard API to differentiate between integrated and discrete GPU devices.
DPCT1007	Migration of *<API name>* is not supported.
DPCT1008	*<function name>* function is not defined in SYCL. This is a hardware-specific feature. Consult with your hardware vendor to find a replacement.
DPCT1009	SYCL uses exceptions to report errors and does not use the error codes. The original code was commented out and a warning string was inserted. You need to rewrite this code.
DPCT1010	SYCL uses exceptions to report errors and does not use the error codes. The call was replaced with 0. You need to rewrite this code.
DPCT1011	The tool detected overloaded operators for built-in vector types, which may conflict with the SYCL 2020 standard operators (see 4.14.2.1 Vec interface). The tool inserted a namespace to avoid the conflict. Use SYCL 2020 standard operators instead.
DPCT1012	Detected kernel execution time measurement pattern and generated an initial code for time measurements in SYCL. You can change the way time is measured depending on your goals.
DPCT1013	The rounding mode could not be specified and the generated code may have different accuracy than the original code. Verify the correctness. SYCL math built-in function rounding mode is aligned with OpenCL C 1.2 standard.
DPCT1014	The flag and priority options are not supported for SYCL queues. The output parameter(s) are set to 0.
DPCT1015	Output needs adjustment.
DPCT1016	REMOVED. The *<API name>* was not migrated, because parameter(s) *<parameter name a>* and/or *<parameter name b>* could not be evaluated, or because *<parameter name a>* is not equal to *<parameter name b>*. Rewrite this code manually.

ID	Message
DPCT1017	The <*SYCL API name*> call is used instead of the <*CUDA API name*> call. These two calls do not provide exactly the same functionality. Check the potential precision and/or performance issues for the generated code.
DPCT1018	The <*API name*> was migrated, but due to <*reason*>, the generated code performance may be sub-optimal.
DPCT1019	`local_mem_size` in SYCL is not a complete equivalent of <*variable name*> in CUDA. You may need to adjust the code.
DPCT1020	Migration of <*api name*>, if it is called from `__global__` or `__device__` function, is not supported. You may need to redesign the code to use the host-side <*api name*> instead, which submits this call to the SYCL queue automatically.
DPCT1021	Migration of `cublasHandle_t` in `__global__` or `__device__` function is not supported. You may need to redesign the code.
DPCT1022	There is no exact match between the `maxGridSize` and the `max_nd_range` size. Verify the correctness of the code.
DPCT1023	The SYCL sub-group does not support mask options for <*API name*>. <*Suggested fix*>.
DPCT1024	The original code returned the error code that was further consumed by the program logic. This original code was replaced with 0. You may need to rewrite the program logic consuming the error code.
DPCT1025	The SYCL queue is created ignoring the flag and priority options.
DPCT1026	The call to <*API name*> was removed because <*reason*>.
DPCT1027	The call to <*API name*> was replaced with 0 because <*reason*>.
DPCT1028	The <*API name*> was not migrated because <*reason*>.
DPCT1029	SYCL currently does not support getting device resource limits. The output parameter(s) are set to 0.
DPCT1030	SYCL currently does not support inter-process communication (IPC) operations. You may need to rewrite the code.
DPCT1031	Memory access across peer devices is an implementation-specific feature which may not be supported by some SYCL backends and compilers. The output parameter(s) are set to 0. You can migrate the code with peer access extension if you do not specify -no-dpcpp-extensions=peer_access.
DPCT1032	A different random number generator is used. You may need to adjust the code.
DPCT1033	Migrated code uses a basic Sobol generator. Initialize `oneapi::mkl::rng::sobol` generator with user-defined direction numbers to use it as Scrambled Sobol generator.
DPCT1034	Migrated API does not return an error code. 0 is returned in the lambda. You may need to rewrite this code.
DPCT1035	All SYCL devices can be used by the host to submit tasks. You may need to adjust this code.
DPCT1036	REMOVED. The type <*type name*> was not migrated because <*reason*>.
DPCT1037	Rewrite this code using <*library name*> with SYCL.

ID	Message
DPCT1038	When the kernel function name is used as a macro argument, the migration result may be incorrect. You need to verify the definition of the macro.
DPCT1039	The generated code assumes that <*parameter name*> points to the global memory address space. If it points to a local memory address space, replace <*function name*> with <*function name*>.
DPCT1040	Use `sycl::stream` instead of `printf` if your code is used on the device.
DPCT1041	SYCL uses exceptions to report errors, it does not use error codes. 0 is used instead of an error code in <*statement name*> statement. You may need to rewrite this code.
DPCT1042	The size of the arguments passed to the SYCL kernel exceeds the minimum size limit (1024) for a non-custom SYCL device. You can get the hardware argument size limit by querying `info::device::max_parameter_size`. You may need to rewrite this code if the size of the arguments exceeds the hardware limit.
DPCT1043	The version-related API is different in SYCL. An initial code was generated, but you need to adjust it.
DPCT1044	<*BaseClass1*> was removed because <*BaseClass2*> has been deprecated in C++11. You may need to remove references to typedefs from <*BaseClass1*> in the class definition.
DPCT1045	Migration is only supported for this API for the <matrix type> sparse matrix type. You may need to adjust the code.
DPCT1046	REMOVED. The <*original API name*> was not migrated because <*reason*>. You need to adjust the code.
DPCT1047	The meaning of <*parameter name*> in the <*API name*> is different from the <*API name*>. You may need to check the migrated code.
DPCT1048	The original value <*macro name*> is not meaningful in the migrated code and was removed or replaced with 0. You may need to check the migrated code.
DPCT1049	The work-group size passed to the SYCL kernel may exceed the limit. To get the device limit, query `info::device::max_work_group_size`. Adjust the work-group size if needed.
DPCT1050	The template argument of the <*type*> could not be deduced. You need to update this code.
DPCT1051	SYCL does not support a device property functionally compatible with <*property name*>. It was migrated to <*migrated token*>. You may need to adjust the value of <*migrated token*> for the specific device.
DPCT1052	SYCL does not support the member access for a volatile qualified vector type. The volatile qualifier was removed. You may need to rewrite the code.
DPCT1053	Migration of device assembly code is not supported.
DPCT1054	The type of variable <*variable name*> is declared in device function with the name <*type*>. Adjust the code to make the <*type*> declaration visible at the accessor declaration point.
DPCT1055	Vector types with size 1 are migrated to the corresponding fundamental types, which cannot be inherited. You need to rewrite the code.
DPCT1056	The use of <*variable name*> in device code was not detected. If this variable is also used in device code, you need to rewrite the code.

ID	Message
DPCT1057	Variable <*variable name*> was used in host code and device code. <*variable name*> type was updated to be used in SYCL device code and new <*host variable name*> was generated to be used in host code. You need to update the host code manually to use the new <*host variable name*>.
DPCT1058	<*API Name*> is not migrated because it is not called in the code.
DPCT1059	SYCL only supports 4-channel image format. Adjust the code.
DPCT1060	SYCL range can only be a 1D, 2D, or 3D vector. Adjust the code.
DPCT1061	Call to <*macro name*> macro was removed because it only contains code, which is unnecessary in SYCL.
DPCT1062	SYCL Image doesn't support normalized read mode.
DPCT1063	Advice parameter is device-defined <*additional action*>. You may need to adjust it.
DPCT1064	Migrated <function name> call is used in a macro/template definition and may not be valid for all macro/template uses. Adjust the code.
DPCT1065	Consider replacing `sycl::<...>::barrier()` with `sycl::<...>::barrier(sycl::access::fence_space::local_space)` for better performance if there is no access to global memory.
DPCT1066	REMOVED. Input/Output distance are migrated to FWD/BWD_DISTANCE. You may need to update the code manually.
DPCT1067	The <*parameter name*> parameter could not be migrated. You may need to update the code manually.
DPCT1068	REMOVED. The value of <*argument type*> could not be deduced. You need to update <*placeholder*> manually.
DPCT1069	The argument <*argument name*> of the kernel function contains virtual pointer(s), which cannot be dereferenced. Try to migrate the code with `usm-level=restricted`.
DPCT1070	<*pointer variable name*> is allocated by `dpct::dpct_malloc`. Use `dpct::get_host_ptr<type>(pointer variable name)` to access the pointer from the host code.
DPCT1071	REMOVED. The placement of the FFT computational function could not be deduced, so it is assumed out-of-place. You may need to adjust the code.
DPCT1072	SYCL currently does not support getting the available memory on the current device. You may need to adjust the code.
DPCT1073	The field values of parameter <*parameter name*> could not be deduced, so the call was not migrated. You need to update this code manually.
DPCT1074	The SYCL Image class does not support some of the flags used in the original code. Unsupported flags were ignored. Data read from SYCL Image could not be normalized as specified in the original code.
DPCT1075	REMOVED. Migration of cuFFT calls may be incorrect and require review.

ID	Message
DPCT1076	The device attribute was not recognized. You may need to adjust the code.
DPCT1077	*<macro name>* redefines a standard SYCL type, which may cause conflicts.
DPCT1078	Consider replacing `memory_order::acq_rel` with `memory_order::seq_cst` for correctness if strong memory order restrictions are needed.
DPCT1079	REMOVED. Replace "dpct_placeholder" with the file path of the dynamic library.
DPCT1080	Variadic functions cannot be called in a SYCL kernel or by functions called by the kernel. You may need to adjust the code.
DPCT1081	The generated code assumes that *<pointer variable>* points to the global memory address space. If it points to a local or private memory address space, replace `address_space::global` with `address_space::local` or `address_space::private`.
DPCT1082	Migration of *<type name>* type is not supported.
DPCT1083	The size of *<placeholder>* in the migrated code may be different from the original code. Check that the allocated memory size in the migrated code is correct.
DPCT1084	The function call *<function name>* has multiple migration results in different template instantiations that could not be unified. You may need to adjust the code.
DPCT1085	The function *<function name>* requires sub-group size to be *<size>*, while other sub-group functions in the same SYCL kernel require a different sub-group size. You may need to adjust the code.
DPCT1086	`__activemask()` is migrated to `0xffffffff`. You may need to adjust the code.
DPCT1087 [UPDATE]	SYCL currently does not support cross group synchronization. You can specify `--use-experimental-features=root-group` to use the root-group to migrate *<synchronization API call>*.
DPCT1088	The macro definition has multiple migration results in the dimension of free queries function that could not be unified. You may need to modify the code.
DPCT1089	The value of the sub-group size attribute argument *<argument name>* cannot be evaluated. Replace `dpct_placeholder` with an integral constant expression.
DPCT1090	SYCL does not support the device property that would be functionally compatible with *<property name>*. It was not migrated. You need to rewrite the code.
DPCT1091	The function `dpct::segmented_reduce` only supports DPC++ native binary operation. Replace "dpct_placeholder" with a DPC++ native binary operation.
DPCT1092	Consider replacing work-group size 128 with different value for specific hardware for better performance.
DPCT1093	The *<Device ID>* device may be not the one intended for use. Adjust the selected device if needed.
DPCT1094	Content contains misleading bidirectional Unicode characters.
DPCT1095	The identifier *<Identifier A>* is confusable with another identifier *<Identifier B>*.

ID	Message
DPCT1096	The right-most dimension of the work-group used in the SYCL kernel that calls this function may be less than <*value of kernel sub-group size attribute*>. The function <*help function name*> may return an unexpected result on the CPU device. Modify the size of the work-group to ensure that the value of the right-most dimension is a multiple of <*value of kernel sub-group size attribute*>.
DPCT1097	The function <*backward function name*> may require the workspace used to save intermediate results from function <*forward function name*>. By default, a workspace from `engine_ext` is selected according to the source data pointer, but this may be incorrect and cause a workspace data race. You may need to rewrite this code.
DPCT1098	The <*expression text*> expression is used instead of the <*function name*> call. These two expressions do not provide the exact same functionality. Check the generated code for potential precision and/or performance issues.
DPCT1099	Verify if the default value of the direction and placement used in the function <*function name*> aligns with the related `compute` function call.
DPCT1100	Currently the DFT external workspace feature in the Intel® oneAPI Math Kernel Library (oneMKL) is only supported on GPU devices. Use the internal workspace if your code should run on non-GPU devices.
DPCT1101	<*expression text*> expression was replaced with a value. Modify the code to use the original expression, provided in comments, if it is correct.
DPCT1102	Zero-length arrays are not permitted in SYCL device code.
DPCT1103	<*expression text*> should be a dynamic library. The dynamic library should supply wrapped kernel functions.
DPCT1104	<*expression text*> should point to a dynamic library loaded in memory. The dynamic library should supply wrapped kernel functions.
DPCT1105	The mcg59 random number generator is used. The subsequence argument <expression> is ignored. You need to verify the migration.
DPCT1106	<*function name*> was migrated with the Intel extensions for device information which may not be supported by all compilers or runtimes. You may need to adjust the code.
DPCT1107	Migration for this overload of <API Name> is not supported.
DPCT1108	<original API> was migrated with the experimental feature <feature name> which may not be supported by all compilers or runtimes. You may need to adjust the code.
DPCT1109	<language feature> cannot be called in SYCL device code. You need to adjust the code.
DPCT1110	The total declared local variable size in device function <function name> exceeds 128 bytes and may cause high register pressure. Consult with your hardware vendor to find the total register size available and adjust the code, or use smaller sub-group size to avoid high register pressure.
DPCT1111	Please verify the input arguments of <migrated API> base on the target function <kernel function name>.
DPCT1112	If the filter mode is set to **linear**, the behavior of image "read" may be different from **tex1Dfetch** in the original code. You may need to adjust the code.
DPCT1113	Consider replacing `sycl::nd_item::barrier(sycl::access::fence_space::local_space)` with `sycl::nd_item::barrier()` if function <*function name*> is called in a multidimensional kernel.

ID	Message
DPCT1114	`cudaMemcpy` is migrated to asynchronization `memcpy`, assuming in the original code the source host memory is pageable memory. If the memory is not pageable, call `wait()` on event return by `memcpy` API to ensure synchronization behavior.
DPCT1115	The `sycl::ext::oneapi::group_local_memory_for_overwrite` is used to allocate group-local memory at the none kernel functor scope of a work-group data parallel kernel. You may need to adjust the code.
DPCT1116	The <*original API*> was migrated to <*migrated API*> for performance, and <*expression*> is computed by `(UINT_MAX + 1) / ('<argument>' + 1)`. This migration requires the initial value of <*value*> to be scaled by multiplying <*expression*>, and any usage of value of <*value*> outside atomic function to be scaled by dividing <*expression*>.
DPCT1117	There is `atomicInc/Dec` operation on <*argument 2*> and value of <*argument 2*> was scaled by migration for performance, refer to DPCT1116. Using the value of <*argument 2*> through <*argument 1*> should also be scaled by dividing or multiplying <*expression*>. You may need to adjust the code.
DPCT1118	SYCL group functions and algorithms must be encountered in converged control flow.

DPCT1000

Message

Error handling `if-stmt` was detected but could not be rewritten.

Detailed Help

This warning is generated when Intel® DPC++ Compatibility Tool detects more complex error handling than it considers safe to remove.

The CUDA* API returns error codes that are consumed by the program logic. SYCL* uses exceptions to report errors and does not return the error code.

When the error handling logic in the original code is simple (for example, a print error message and exit), the code is removed in the resulting Data Parallel C++ (DPC++) application. The expectation is that SYCL throws an exception, which is handled with the printing of an exception message and exiting (the exception handler is generated automatically by Intel® DPC++ Compatibility Tool).

Suggestions to Fix

Review the error handling `if-` statement and try to rewrite it to use an exception handler instead.

For example, this original CUDA code:

```
void check_err(cudaError_t err, float **f) {
  if (err != cudaSuccess) {
   log_error();
  }
}
```

```
void foo() {
  float *f;
  cudaError_t err = cudaMalloc(&f, 4);
  check_err(err, &f);
}
```

results in the following migrated SYCL code:

```
void check_err(int err, float **f) {
  /*
  DPCT1000:1: Error handling if-stmt was detected but could not be rewritten.
  */
  if (err != 0) {
    /*
    DPCT1001:0: The statement could not be removed.
    */
    log_error();
  }
}

void foo() {
  float *f;
  ...
  int err = (f = (float *)sycl::malloc_device(4, dpct::get_default_queue()), 0);
  check_err(err, &f);
}
```

which is rewritten to:

```
void foo() {
  float *f;
  try {
   f = (float *)sycl::malloc_device(4, dpct::get_default_queue());
  } catch (sycl::exception const &e) {
    log_error();
  }
}
```

DPCT1001

Message

The statement could not be removed.

Detailed Help

Intel® DPC++ Compatibility Tool was not able to remove the code in the `then` clause of `if-stmt`. See DPCT1000.

Suggestions to Fix

Refer to DPCT1000.

DPCT1002

Message

Special case error handling `if-stmt` was detected. You may need to rewrite this code.

Detailed Help

See DPCT1000.

Suggestions to Fix

For example, this original CUDA* code:

```
void foo() {
  float *f;
  cudaError_t err = cudaMalloc(&f, 4);
  if (err == cudaErrorMemoryAllocation) {
    std::cerr << "cudaErrorMemoryAllocation" << std::endl;
  }
}
```

results in the following migrated SYCL* code:

```
void foo() {
  float *f;
  int err = (f = (float *)sycl::malloc_device(4, dpct::get_default_queue()), 0);
  /*
  DPCT1002:1: Special case error handling if-stmt was detected. You may need to
  rewrite this code.
  */
  if (err == 2) {
    std::cerr << "cudaErrorMemoryAllocation" << std::endl;
  }
}
```

which is rewritten to:

```
void foo() {
  float *f;
  try {
    f = (float *)sycl::malloc_device(4, dpct::get_default_queue());
  } catch (sycl::exception const &e) {
    std::cerr << e.what() << std::endl;
  }
}
```

DPCT1003

Message

Migrated API does not return error code. `(*, 0)` is inserted. You may need to rewrite this code.

Detailed Help

Typically, this happens because the CUDA* API returns an error code and then it is consumed by the program logic.

SYCL* uses exceptions to report errors and does not return the error code.

Intel® DPC++ Compatibility Tool inserts a `(*, 0)` operator, so that the resulting application can be compiled. This operator returns 0 and is inserted if the return code is expected by the program logic and the new API does not return it. You should review all such places in the code.

Suggestions to Fix

If in a DPC++ application you:

- Do not need the code that consumes the error code, remove the code and the `(*, 0)` operator.
- Need the code that consumes the error code, try to replace it with an exception handling code and use your logic in an exception handler.

For example, this original CUDA code:

```
void foo() {
  cudaError_t err;
  float *f;
  err = cudaMalloc(&f, 4);
  err = cudaFree(f);
}
```

results in the following migrated SYCL code:

```
void foo() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
  int err;
  float *f;
  /*
  DPCT1003:0: Migrated API does not return error code. (*, 0) is inserted. You
  may need to rewrite this code.
  */
  err = (f = (float *)sycl::malloc_device(4, q_ct1), 0);
  /*
  DPCT1003:1: Migrated API does not return error code. (*, 0) is inserted. You
  may need to rewrite this code.
```

```
  */
  err = (sycl::free(f, q_ct1), 0);
}
```

which is rewritten to:

```
void foo() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();

  float *f;
  f = (float *)sycl::malloc_device(4, q_ct1);
  sycl::free(f, q_ct1);
}
```

DPCT1004

NOTE This diagnostic message is no longer generated by Intel® DPC++ Compatibility Tool 2023.0 or later.

Message

Compatible SYCL code could not be generated.

Detailed Help

Visit the Migrating to SYCL forum for additional help.

Suggestions to Fix

Visit the Migrating to SYCL forum for additional help.

DPCT1005

Message

The SYCL device version is different from CUDA Compute Compatibility. You may need to rewrite this code.

Detailed Help

Intel® DPC++ Compatibility Tool detected a usage of CUDA* Compute Capability-dependent logic in the original program.

DPC++ does not support CUDA Compute Capability. The logic in the generated code uses a version extracted from the `cl::sycl::info::device::version`, which is different than CUDA Compute Capability.

Suggestions to Fix

Review the logic and adjust it.

For example, this original CUDA code:

```
int main() {
  cudaDeviceProp deviceProp;
  int deviceID = 0;
  cudaGetDeviceProperties(&deviceProp, deviceID);
  if (deviceProp.major >= 3) {
    // Use functionality from Compute Capability 3.0+
  }
  return 0;
}
```

results in the following migrated SYCL* code:

```
int main() {
  dpct::device_info deviceProp;
  dpct::dev_mgr::instance().get_device(0).get_device_info(deviceProp);
  /*
  DPCT1005:0: The SYCL device version is different from CUDA Compute
  Compatibility. You may need to rewrite this code.
  */
  if (deviceProp.get_major_version() >= 3) {
    // Invoke kernel using __shfl() which requires Compute Capability 3.0+
  }
  return 0;
}
```

which is rewritten to:

```
int main() {
  // sycl::select_from_group is supported in SYCL 2020.
  if (SYCL_LANGUAGE_VERSION >= 202000) {
    // Invoke kernel using sycl::select_from_group
    ...
  }
  return 0;
}
```

DPCT1006

Message

SYCL does not provide a standard API to differentiate between integrated and discrete GPU devices.

Detailed Help

The generated code uses a `0` (zero) as the value of this field.

Suggestions to Fix

Consider re-implementing the code that depends on this field.

For example, this original CUDA* code:

```
void foo() {
  int device_number = 0;
    cudaDeviceProp prop;
    cudaGetDeviceProperties(&prop, device_number);
    if (prop.integrated) {
      ...
    } else {
      ...
    }
}
```

results in the following migrated SYCL* code:

```
void foo() {
  int device_number = 0;
    dpct::device_info prop;
    dpct::dev_mgr::instance().get_device(device_number).get_device_info(prop);
    /*
    DPCT1006:0: SYCL does not provide a standard API to differentiate between
    integrated and discrete GPU devices.
    */
    if (prop.get_integrated()) {
      ...
    } else {
      ...
    }
  }
```

which is rewritten to:

```
void foo() {
  int device_number = 0;
    dpct::device_info prop;
    // 1. User can use device name or other info to select device. The `a_device_name` can be
obtained by running command `   sycl-ls`
    // 2. User can use the environment variable `ONEAPI_DEVICE_SELECTOR` to filter out some
devices.
    // 3. User can use the user-provided device-selector like:
    //   int prefer_my_device(const sycl::device& d) {
    //     return d.get_info<info::device::name>() == "MyDevice";
    //   }
    //   sycl::device preferred_device { prefer_my_device };
    if (dpct::dev_mgr::instance().get_device(device_numbe).get_info<sycl::info::device::name>()
== "a_device_name") {
      ...
    } else {
      ...
    }
  }
```

DPCT1007

Message

Migration of *<API name>* is not supported.

Detailed Help

Visit the Migrating to SYCL forum for additional help.

Suggestions to Fix

This kind of code requires manual migration to SYCL*.

DPCT1008

Message

<function name> function is not defined in SYCL. This is a hardware-specific feature. Consult with your hardware vendor to find a replacement.

Detailed Help

The clock function call was not replaced, because it is not defined in SYCL*. Replace this hardware-specific feature with the one offered by your hardware vendor.

Suggestions to Fix

Consult with your hardware vendor to find a replacement.

For example, this original CUDA* code:

```
__global__ void k(clock_t *timer) {
  int tid = threadIdx.x;
  if (tid == 0)
    timer[0] = clock();

  // workload

  __syncthreads();
  if (tid == 0)
```

```
    timer[1] = clock();
}

int main() {
  clock_t *dtimer = NULL;
  cudaMalloc((void **)&dtimer, sizeof(clock_t) * 2);
  clock_t timer[2];
  k<<<1, 128>>>(dtimer);
  cudaMemcpy(timer, dtimer, sizeof(clock_t) * 2, cudaMemcpyDeviceToHost);
  cudaFree(dtimer);
  long double time = timer[1] - timer[0];
  return 0;
}
```

results in the following migrated SYCL code:

```
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
#include <time.h>
void k(clock_t *timer, sycl::nd_item<3> item_ct1) {
  int tid = item_ct1.get_local_id(2);
  if (tid == 0)
    /*
    DPCT1008:1: clock function is not defined in SYCL. This is a
    hardware-specific feature. Consult with your hardware vendor to find a
    replacement.
    */
    timer[0] = clock();// clock() is used to measure the kernel runtime

  // workload

  item_ct1.barrier();
  if (tid == 0)
    /*
             DPCT1008:2: clock function is not defined in SYCL. This is a
             hardware-specific feature. Consult with your hardware vendor to find a
     replacement.
    */
    timer[1] = clock();
}

int main() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
  clock_t *dtimer = NULL;
  dtimer = sycl::malloc_device<clock_t>(2, q_ct1);
  clock_t timer[2];
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 128), sycl::range<3>(1, 1, 128)),
      [=](sycl::nd_item<3> item_ct1) {
        k(dtimer, item_ct1);
      });
  q_ct1.memcpy(timer, dtimer, sizeof(clock_t) * 2).wait();
  sycl::free(dtimer, q_ct1);
  long double time = timer[1] - timer[0];
  return 0;
}
```

which is rewritten to:

```
#define DPCT_PROFILING_ENABLED
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
#include <time.h>
void k(sycl::nd_item<3> item_ct1) {
  // workload
}

int main() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
  dpct::event_ptr start;
  dpct::event_ptr end;
  start = new sycl::event();
  end = new sycl::event();
  *start = q_ct1.ext_oneapi_submit_barrier();
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 128), sycl::range<3>(1, 1, 128)),
      [=](sycl::nd_item<3> item_ct1) {
        k(item_ct1);
      });
  *end = q_ct1.ext_oneapi_submit_barrier();
  end->wait_and_throw();
  long double time =
      (end->get_profiling_info<sycl::info::event_profiling::command_end>() -
       start
           ->get_profiling_info<sycl::info::event_profiling::command_start>()) /
      1000000.0f;  return 0;
}
```

DPCT1009

Message

SYCL uses exceptions to report errors and does not use the error codes. The original code was commented out and a warning string was inserted. You need to rewrite this code.

Detailed Help

SYCL* uses exceptions to report errors and does not use error codes. The original code tries to get a string message through the error code, while SYCL does not require such functionality.

To indicate that the code needs to be updated, a warning string is inserted.

Suggestions to Fix

You may need to rewrite this code.

For example, this original CUDA* code:

```
void foo() {
  float *f;
  cudaError_t err = cudaMalloc(&f, 4);
  printf("%s\n", cudaGetErrorString(err));
}
```

results in the following migrated SYCL code:

```
void foo() {
  float *f;
  int err = (f = (float *)sycl::malloc_device(4, dpct::get_default_queue()), 0);
  /*
  DPCT1009:1: SYCL uses exceptions to report errors and does not use the error
  codes. The original code was commented out and a warning string was inserted.
  You need to rewrite this code.
  */
  printf("%s\n",
         "cudaGetErrorString is not supported" /*cudaGetErrorString(err)*/);
}
```

which is rewritten to:

```
void foo() {
  float *f;
  try {
    f = (float *)sycl::malloc_device(4, dpct::get_default_queue())
  } catch (sycl::exception const &e) {
    std::cerr << e.what() << std::endl;
  }
}
```

DPCT1010

Message

SYCL uses exceptions to report errors and does not use the error codes. The call was replaced with 0. You need to rewrite this code.

Detailed Help

SYCL* uses exceptions to report errors and does not use error codes. The original code tries to get an error code, while SYCL does not require such functionality. The call was replaced with 0.

Suggestions to Fix

You may need to rewrite this code.

For example, this original CUDA* code:

```
__global__ void kernel() {
  ...
}

void foo() {
  kernel<<<1, 1>>>();
  cudaDeviceSynchronize();
  cudaError_t err = cudaGetLastError();
  printf("%d\n", err);
}
```

results in the following migrated SYCL code:

```
void kernel() {
  ...
}

void foo() {
  dpct::get_default_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel();
      });
  dpct::get_current_device().queues_wait_and_throw();
  /*
  DPCT1010:0: SYCL uses exceptions to report errors and does not use the error
  codes. The call was replaced with 0. You need to rewrite this code.
  */
  int err = 0;
  printf("%d\n", err);
}
```

which is rewritten to:

```
void kernel() {
  ...
}

void foo() {
  try {
    dpct::get_default_queue().parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel();
        });
    dpct::get_current_device().queues_wait_and_throw();
  } catch (sycl::exception const &e) {
    std::cerr << e.what() << std::endl;
  }
}
```

DPCT1011

Message

The tool detected overloaded operators for built-in vector types, which may conflict with the SYCL 2020 standard operators (see 4.14.2.1 Vec interface). The tool inserted a namespace to avoid the conflict. Use SYCL 2020 standard operators instead.

Detailed Help

You may have overloaded operators for vector types such as `double2`. This resulted in a conflict because overloaded operators with the same signature are also defined in the SYCL* standard. Intel® DPC++ Compatibility Tool adds the namespace for overloaded operators to differentiate them from the ones defined in SYCL. You may need to rewrite the code.

Suggestions to Fix

You may need to rewrite this code.

For example, this original CUDA* code:

```
__host__ __device__ double2 &operator+=(double2 &a, const double2 &b) {
  a.x += b.x;
  a.y += b.y;
  return a;
}

void foo(double2 &a, double2 &b) {
  a += b;
}
```

results in the following migrated SYCL code:

```
/*
DPCT1011:0: The tool detected overloaded operators for built-in vector types,
which may conflict with the SYCL 2020 standard operators (see 4.14.2.1 Vec
interface). The tool inserted a namespace to avoid the conflict. Use SYCL 2020
standard operators instead.
*/
namespace dpct_operator_overloading {
  sycl::double2 &operator+=(sycl::double2 &a, const sycl::double2 &b) {
    a.x() += b.x();
    a.y() += b.y();
    return a;
  }
} // namespace dpct_operator_overloading

void foo(sycl::double2 &a, sycl::double2 &b) {
  dpct_operator_overloading::operator+=(a, b);
}
```

which is rewritten to:

```
void foo(sycl::double2 &a, sycl::double2 &b) {
  // In this case, the user-defined overloading of `+=` has been supported by sycl::double2, so
we can use the operator `+=` directly.
  a += b;
}
```

DPCT1012

Message

Detected kernel execution time measurement pattern and generated an initial code for time measurements in SYCL. You can change the way time is measured depending on your goals.

Detailed Help

The generated code uses the CPU time to measure the kernel execution time. You can change the way time is measured depending on your requirements.

Suggestions to Fix

Review the logic and adjust it as needed.

For example, this original CUDA* code:

```
__global__ void kernel() {
  ...
}

void foo() {
  cudaEvent_t start;
  cudaEvent_t end;
  cudaEventCreate(&start);
  cudaEventCreate(&end);
  cudaEventRecord(start);
  kernel<<<1, 1>>>();
  cudaEventRecord(end, 0);
  cudaEventSynchronize(end);
  float time;
  cudaEventElapsedTime(&time, start, end);
}
```

results in the following migrated SYCL* code:

```
void kernel() {
  ...
}

void foo() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
```

```
  dpct::event_ptr start;
  std::chrono::time_point<std::chrono::steady_clock> start_ct1;
  dpct::event_ptr end;
  std::chrono::time_point<std::chrono::steady_clock> end_ct1;
  start = new sycl::event();
  end = new sycl::event();
  /*
  DPCT1012:0: Detected kernel execution time measurement pattern and generated
  an initial code for time measurements in SYCL. You can change the way time is
  measured depending on your goals.
  */
  start_ct1 = std::chrono::steady_clock::now();
  *end = q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel();
      });
  /*
  DPCT1012:1: Detected kernel execution time measurement pattern and generated
  an initial code for time measurements in SYCL. You can change the way time is
  measured depending on your goals.
  */
  end->wait();
  end_ct1 = std::chrono::steady_clock::now();
  float time;
  time = std::chrono::duration<float, std::milli>(end_ct1 - start_ct1).count();
}
```

which is rewritten to:

```
// User can add `--enable-profiling` option to migrate the code
void kernel() {
  ...
}

void foo() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
  dpct::event_ptr start;
  dpct::event_ptr end;
  start = new sycl::event();
  end = new sycl::event();
  *start = q_ct1.ext_oneapi_submit_barrier();
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel();
      });
  *end = q_ct1.ext_oneapi_submit_barrier();
  end->wait_and_throw();
  float time;
  time =
      (end->get_profiling_info<sycl::info::event_profiling::command_end>() -
       start
           ->get_profiling_info<sycl::info::event_profiling::command_start>()) /
      1000000.0f;
}
```

DPCT1013

Message

The rounding mode could not be specified and the generated code may have different accuracy than the original code. Verify the correctness. SYCL math built-in function rounding mode is aligned with OpenCL C 1.2 standard.

Detailed Help

The rounding mode could not be specified and the generated code may have different accuracy than the original code. Verify the correctness. The SYCL* math built-in functions rounding mode is aligned with the OpenCL C 1.2 standard.

Suggestions to Fix

Review the logic and adjust it.

For example, this original CUDA* code:

```
__global__ void foo(double *res) {
  double a = 1.00001;
  double b = 100000000000000000;
  *res = __dadd_ru(a, b);
}
```

results in the following migrated SYCL code:

```
void foo(double *res) {
  double a = 1.00001;
  double b = 100000000000000000;
  /*
  DPCT1013:0: The rounding mode could not be specified and the generated code
  may have different accuracy than the original code. Verify the correctness.
  SYCL math built-in function rounding mode is aligned with OpenCL C 1.2
  standard.
  */
  *res = a + b;
}
```

which is rewritten to:

```
void foo(double *res) {
  double a = 1.00001;
  double b = 100000000000000000;
  *res = a + b; // For precision, use: *res = sycl::nextafter(a + b, (double)INFINITY);

}
```

DPCT1014

Message

The flag and priority options are not supported for SYCL queues. The output parameter(s) are set to 0.

Detailed Help

The flag and priority options are not supported in the SYCL* queue. You may need to rewrite the generated code.

Suggestions to Fix

Review the logic and adjust it.

For example, this original CUDA* code:

```
void foo(cudaStream_t old_stream, cudaStream_t *new_stream) {
  unsigned int flag;
  cudaStreamGetFlags(old_stream, &flag);
  cudaStreamCreateWithFlags(new_stream, flag);
}
```

results in the following migrated SYCL code:

```
void foo(sycl::queue *old_stream, sycl::queue **new_stream) {
  unsigned int flag;
  /*
  DPCT1014:0: The flag and priority options are not supported for SYCL queues.
  The output parameter(s) are set to 0.
  */
  *(&flag) = 0;
  /*
  DPCT1025:1: The SYCL queue is created ignoring the flag and priority options.
  */
  *(new_stream) = dpct::get_current_device().create_queue();
}
```

which is rewritten to:

```
void foo(sycl::queue *old_stream, sycl::queue **new_stream) {
  *(new_stream) = dpct::get_current_device().create_queue();
}
```

DPCT1015

Message

Output needs adjustment.

Detailed Help

SYCL* device code uses SYCL output stream instead of `printf`-like formatted strings. Adjustments may be needed to achieve the desired output.

Suggestions to Fix

Use a SYCL output stream instead of a `printf`-like formatted string.

For example, this original CUDA* code:

```
__global__ void kernel() {
  ...
  printf("value is %d\n", value);
}
```

results in the following migrated SYCL code:

```
void kernel(const sycl::stream &stream_ct1) {
  ...
  /*
  DPCT1015:0: Output needs adjustment.
  */
  stream_ct1 << "value is %d\n";
}
```

which is rewritten to:

```
void kernel(const sycl::stream &stream_ct1) {
  ...
  stream_ct1 << "value is " << value << "\n";
}
```

DPCT1016

NOTE This diagnostic message is no longer generated by Intel® DPC++ Compatibility Tool 2023.0 or later.

Message

The *<API name>* was not migrated, because parameter(s) *<parameter name a>* and/or *<parameter name b>* could not be evaluated, or because *<parameter name a>* is not equal to *<parameter name b>*. Rewrite this code manually.

Detailed Help

The `cublasSetMatrix` can be migrated by Intel® DPC++ Compatibility Tool only when `lda` and `ldb` have the same constant value. In cases where the migration did not occur, rewrite the code manually.

- If the values of `lda` and `ldb` are the same, you can use the following code:

```
dpct::dpct_memcpy((void*)(B), (void*)(A), lda*cols*elemSize, dpct::host_to_device); // For
cublasGetMatrix, use dpct::device_to_host
```

- Otherwise, you can copy the columns of the matrix or elements of the vector one by one.

- The equivalent of `cublasSetMatrix(rows, cols, elemSize, A, lda, B, ldb)` is:

```
auto A_backup = A;
auto B_backup = B;
A = A - lda;
B = B - ldb;
for (int c = 0; c < cols; ++c) {
  A = A + lda;
  B = B + ldb;
  dpct::dpct_memcpy(
      (void *)(B), (void *)(A), rows * elemSize,
      dpct::host_to_device); // For cublasGetMatrix, use dpct::device_to_host
}
A = A_backup;
B = B_backup;
```

Intel® DPC++ Compatibility Tool can migrate the `cublasSetVector` only when `incx` and `incy` have the same constant value. In cases where the migration does not occur, rewrite the code manually.

- If the values of `incx` and `incy` are the same, you can use the following code:

```
dpct::dpct_memcpy((void*)(B), (void*)(A), n*incx*elemSize, dpct::host_to_device); // For
cublasGetVector, use dpct::device_to_host
```

- Otherwise, you can copy the elements of the vector one by one:
  - To replace `cublasGetVector(n, elemSize, x, incx, y, incy)`, use the following snippet:

```
auto x_backup = x;
auto y_backup = y;
x = x - incx;
y = y - incy;
for (int c = 0; c < n; ++c) {
  x = x + incx;
  y = y + incy;
  dpct::dpct_memcpy(
      (void *)(y), (void *)(x), elemSize,
      dpct::device_to_host); // for cublasSetVector, use dpct::host_to_device
}
x = x_backup;
y = y_backup;
```

Suggestions to Fix

Review the logic and adjust it.

DPCT1017

Message

The *<SYCL API name>* call is used instead of the *<CUDA API name>* call. These two calls do not provide exactly the same functionality. Check the potential precision and/or performance issues for the generated code.

Detailed Help

Intel® DPC++ Compatibility Tool did not find an exact functional equivalent of the function in SYCL*, so it used the closest alternative. The replacement may impact your code precision and/or performance. Verify the code correctness/performance.

Suggestions to Fix

Verify the code correctness/performance.

For example, this original CUDA* code:

```
__global__ void k(float *out, float *in) {
  *out = normf(3, in);
}
```

results in the following migrated SYCL code:

```
void k(float *out, float *in) {
  /*
  DPCT1017:0: The dpct::length call is used instead of the normf call. These two
  calls do not provide exactly the same functionality. Check the potential
  precision and/or performance issues for the generated code.
  */
  *out = dpct::length(in, 3);
}
```

which is rewritten to:

```
#include <sycl/ext/intel/math.hpp>
void k(float *out, float *in) {
  /*
  If the extension API sycl::ext::intel::math::norm is available, the following
  line can alternately be migrated to *out = sycl::ext::intel::math::norm(3, in);
  */
  *out = dpct::length(in, 3);
}
```

DPCT1018

Message

The <*API name*> was migrated, but due to <*reason*>, the generated code performance may be sub-optimal.

Detailed Help

This warning appears in the following cases:

- Migration of the `cublasSetMatrix` function. Intel® DPC++ Compatibility Tool replaced the `cublasSetMatrix` with memory copying from the host to the device. When the `rows` parameter of the `cublasSetMatrix` is smaller than the `lda` parameter, the generated code copies more data (`lda*cols`) than the actual data available in the matrix (`rows*cols`).

  To improve performance, consider changing the values of `lda` and `ldb`. If the rows parameter is greater than or equal to `lda`, no action is required for this code.
- Migration of the `cublasSetVector` function. Intel® DPC++ Compatibility Tool replaced the `cublasSetVector` with memory copying from the host to the device. When the `incx` parameter of the `cublasSetVector` equals the `incy` parameter, but is greater than 1, the generated code copies more data (`incx*n`) than the actual data available in the vector (`n`). To improve performance, consider changing the values of `incx` and `incy`.

Suggestions to Fix

If the rows parameter of the `cublasSetMatrix` is smaller than the `lda` parameter and you observe performance issues, consider changing the values of `lda` and `ldb`.

If the `incx` parameter of the `cublasSetVector` equals the `incy` parameter, but is greater than 1 and you observe performance issues, consider changing the values of `incx` and `incy`.

For example, this original CUDA* code:

```
void foo() {
  const int element_num = 128;
  const int h_inc = 128;
  const int d_inc = 128;
  cublasSetVector(element_num, sizeof(float), data, h_inc, d_data, d_inc);
}
```

results in the following migrated SYCL* code:

```
void foo() {
  const int element_num = 128;
  const int h_inc = 128;
  const int d_inc = 128;
  /*
  DPCT1018:0: The cublasSetVector was migrated, but due to parameter h_inc
  equals to parameter d_inc but greater than 1, the generated code performance
  may be sub-optimal.
  */
  dpct::matrix_mem_copy((void *)d_data, (void *)data, d_inc, h_inc, 1,
                        element_num, sizeof(float));
}
```

which is rewritten to:

```
void foo() {
  const int element_num = 128;

  //Save the data in d_data continuously and change h_inc and d_inc from 128 to 1.
  const int h_inc = 1;
  const int d_inc = 1;
```

```
  // Now there is no padding between each element, so memcpy can be used directly.
  dpct::get_default_queue().memcpy(d_data, data, sizeof(float) * element_num).wait();
}
```

DPCT1019

Message

`local_mem_size` in SYCL is not a complete equivalent of <*variable name*> in CUDA. You may need to adjust the code.

Detailed Help

In CUDA*, the `sharedMemPerBlock` reports the size of the shared memory in bytes available per block. The SYCL* equivalent of a CUDA block is a work-group. The SYCL equivalent of shared memory is local memory. There is no limitation on the size of the local memory per work-group in SYCL. There is a limit on the maximum size of the local memory in bytes available per compute unit, which is exposed by the `info::device::local_mem_size` device descriptor in SYCL.

Suggestions to Fix

Verify the code correctness.

For example, this original CUDA code:

```
void foo() {
  cudaDeviceProp prop;
  cudaGetDeviceProperties(&prop, 0);
  if (prop.sharedMemPerBlock >= threshold) {
    // submit the task
    Code piece A
  } else {
    // change the block size or block number
    Code piece B
  }
}
```

results in the following migrated SYCL code:

```
void foo() {
  dpct::device_info prop;
  dpct::dev_mgr::instance().get_device(0).get_device_info(prop);
  /*
  DPCT1019:0: local_mem_size in SYCL is not a complete equivalent of
  sharedMemPerBlock in CUDA. You may need to adjust the code.
  */
  if (prop.get_local_mem_size() >= threshold) {
    // submit the task
    Code piece A
```

```
  } else {
    // change the block size or block number
    Code piece B
  }
}
```

which is rewritten to:

```
void foo() {
  dpct::device_info prop;
  dpct::dev_mgr::instance().get_device(0).get_device_info(prop);
  if (prop.get_local_mem_size() >= threshold) {
    // submit the task
    Code piece A
  } else {
    // change the block size or block number
    Code piece B
  }
}
```

DPCT1020

Message

Migration of *<api name>*, if it is called from `__global__` or `__device__` function, is not supported. You may need to redesign the code to use the host-side *<api name>* instead, which submits this call to the SYCL queue automatically.

Detailed Help

The warning message is generated in cases where the *<api name>* itself submits the SYCL* kernel to the command queue, and the caller of *<api-name>* is the SYCL kernel that is submitted to the command queue itself. It results in device-side enqueue of the kernel, which is not supported by SYCL 2020.

Suggestions to Fix

Redesign the code to use the host-side API, which submits this call to the SYCL queue automatically.

For example, this original CUDA* code:

```
__global__ void kernel(float *d_data) {
  int tid = threadIdx.x;
  d_data[tid + 1] = tid;

  __syncthreads();

  if (tid == 0) {
    cublasHandle_t handle;
    cublasCreate(&handle);
    cublasSasum(handle, 128, d_data + 1, 1, d_data)
    cublasDestroy(handle);
```

```
  }
}

void foo() {
  float *d_data;
  cudaMalloc((void **)&d_data, sizeof(float) * (1 + 128));
  kernel<<<1, 128>>>(d_data);

  float data;
  cudaMemcpy(data, d_data, sizeof(float), cudaMemcpyDeviceToHost);
  cudaFree(d_data);
}
```

results in the following migrated SYCL code:

```
void kernel(float *d_data, sycl::nd_item<3> item_ct1) {
  int tid = item_ct1.get_local_id(2);
  d_data[tid + 1] = tid;

  item_ct1.barrier();

  if (tid == 0) {
    /*
    DPCT1021:2: Migration of cublasHandle_t in __global__ or __device__ function
    is not supported. You may need to redesign the code.
    */
    cublasHandle_t handle;
    handle = &dpct::get_default_queue();
    /*
    DPCT1020:1: Migration of cublasSasum, if it is called from __global__ or
    __device__ function, is not supported. You may need to redesign the code to
    use the host-side oneapi::mkl::blas::column_major::asum instead, which submits
    this call to the SYCL queue automatically.
    */
    cublasSasum(handle, 128, d_data + 1, 1, d_data);
    handle = nullptr;
  }
}

void foo() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
  float *d_data;
  d_data = sycl::malloc_device<float>((1 + 128), q_ct1);
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 128), sycl::range<3>(1, 1, 128)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel(d_data, item_ct1);
      });

  float data;
  q_ct1.memcpy(&data, d_data, sizeof(float)).wait();
  sycl::free(d_data, q_ct1);
}
```

which is rewritten to:

```
void kernel(float *d_data, sycl::nd_item<3> item_ct1) {
  int tid = item_ct1.get_local_id(2);
  d_data[tid + 1] = tid;
}

void foo() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
  float *d_data;
  d_data = sycl::malloc_device<float>((1 + 128), q_ct1);
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 128), sycl::range<3>(1, 1, 128)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel(d_data, item_ct1);
      });
  oneapi::mkl::blas::column_major::asum(q_ct1, 128, d_data + 1, 1, d_data);

  float data;
  q_ct1.memcpy(&data, d_data, sizeof(float)).wait();
  sycl::free(d_data, q_ct1);
}
```

DPCT1021

Message

Migration of `cublasHandle_t` in `__global__` or `__device__` function is not supported. You may need to redesign the code.

Detailed Help

The `cublasHandle_t` is migrated to the `cl::sycl::queue`. Because SYCL* 2020 does not support device-side enqueue of the kernel, the usage of the `cl::sycl::queue` is not possible in SYCL kernels.

Suggestions to Fix

Redesign the code to submit all kernels by the host program instead of device-side enqueue.

Refer to DPCT1020.

DPCT1022

Message

There is no exact match between the `maxGridSize` and the `max_nd_range` size. Verify the correctness of the code.

Detailed Help

There is no analogue of the `maxGridSize` in SYCL*. SYCL `nd_ranges` can have up to three dimensions, just like grids in CUDA*, but there is no maximum of `nd_range` size beyond the data type width, which is `size_t`. The Intel® DPC++ Compatibility Tool replaces the `maxGridSize` with the `max_nd_range_size` helper, which is initialized to the `size_t` width.

Suggestions to Fix

Verify the code correctness.

For example, this original CUDA code:

```
void foo() {
  cudaDeviceProp prop;
  cudaGetDeviceProperties(&prop, 0);
  if (prop.maxGridSize[0] >= TASK_SIZE) {
    // submit the task on one kernel
    ...
  } else {
    // split the task into multi kernels
    ...
  }
}
```

results in the following migrated SYCL code:

```
void foo() {
  dpct::device_info prop;
  dpct::dev_mgr::instance().get_device(0).get_device_info(prop);
  /*
  DPCT1022:0: There is no exact match between the maxGridSize and the
  max_nd_range size. Verify the correctness of the code.
  */
  if (prop.get_max_nd_range_size()[0] >= TASK_SIZE) {
    // submit the task on one kernel
    ...
  } else {
    // split the task into multi kernels
    ...
  }
}
```

which is rewritten to:

```
void foo() {
  // submit the task on one kernel
  ...
}
```

DPCT1023

Message

The SYCL sub-group does not support mask options for <*API name*>. <*Suggested fix*>.

Detailed Help

This warning is generated when the SYCL* sub-group function used for migration does not have a mask parameter. This warning can be ignored if the original mask parameter was not used.

Suggestions to Fix

Verify the code correctness.

For example, this original CUDA* code:

```
__device__ void d(unsigned mask, int val, int srcLane) {
  __shfl_sync(mask, val, srcLane);
}
```

results in the following migrated SYCL code:

```
void d(unsigned mask, int val, int srcLane,
       const sycl::nd_item<3> &item_ct1) {
  /*
  DPCT1023:0: The SYCL sub-group does not support mask options for
  dpct::select_from_sub_group. You can specify
  "--use-experimental-features=masked-sub-group-operation" to use the dpct
  experimental helper function to migrate __shfl_sync.
  */
  dpct::select_from_sub_group(item_ct1.get_sub_group(), val, srcLane);
}
```

which is rewritten to:

```
void d(unsigned mask, int val, int srcLane,
       const sycl::nd_item<3> &item_ct1) {
  // If this compiler & runtime support group non-uniform shuffle, then you can
  // use this API
  dpct::experimental::select_from_sub_group(mask, item_ct1.get_sub_group(), val,
                                            srcLane);
}
```

DPCT1024

Message

The original code returned the error code that was further consumed by the program logic. This original code was replaced with 0. You may need to rewrite the program logic consuming the error code.

Detailed Help

This warning is generated in cases where in the original code, the CUDA* API call returns the error code, which is consumed by the program logic:

```
handleError(cudaEventRecord(e));
```

If in the resulting code the CUDA API call is replaced by the code, which does not return the error code, 0 is used as an input to the program logic, consuming the error code:

```
e_ct1 = clock(), handleError(0);
```

The error handling code in that case must be verified and may require replacing it with exception handling code or removed completely.

Suggestions to Fix

Verify the code correctness.

For example, this original CUDA code:

```
void foo() {
  cudaEvent_t e;
  ...
  handleError(cudaEventRecord(e));
  ...
}
```

results in the following migrated SYCL* code:

```
void foo() {
  dpct::event_ptr e;
  std::chrono::time_point<std::chrono::steady_clock> e_ct1;
  ...
  /*
  DPCT1024:1: The original code returned the error code that was further
  consumed by the program logic. This original code was replaced with 0. You may
  need to rewrite the program logic consuming the error code.
  */
  e_ct1 = std::chrono::steady_clock::now();
  handleError(0);
  ...
}
```

which is rewritten to:

```
void foo() {
  dpct::event_ptr e;
  std::chrono::time_point<std::chrono::steady_clock> e_ct1;
  ...
  e_ct1 = std::chrono::steady_clock::now();
  ...
}
```

DPCT1025

Message

The SYCL queue is created ignoring the flag and priority options.

Detailed Help

The flag and priority options are not supported when creating the SYCL* queue. You may need to rewrite the generated code.

Suggestions to Fix

Verify the code correctness.

Refer to DPCT1014.

DPCT1026

Message

The call to <*API name*> was removed because <*reason*>.

Detailed Help

API calls from the original application, which do not have functionally compatible SYCL* API calls are removed if Intel® DPC++ Compatibility Tool determines that it should not affect the program logic.

Possible reasons for removal:

- SYCL currently does not support setting resource limits on devices.
- SYCL currently does not support associating USM with a specific queue.
- SYCL currently does not support query operations on queues.
- SYCL currently does not support capture operations on queues.
- SYCL currently does not support configuring shared memory on devices.
- SYCL currently does not support setting cache config on devices.
- SYCL currently does not support registering of existing host memory for use by device. Use USM to allocate memory for use by host and device.
- SYCL currently does not support setting flags for devices.
- SYCL currently does not support memory access across peer devices.
- There is no corresponding API in SYCL.
- The call is redundant in SYCL.

Suggestions to Fix

Verify the code correctness.

For example, this original CUDA* code:

```
__device__ void bar(int *a) {
  __ldg(a);
}

__global__ void kernel() {
  return;
}

void foo() {
  cudaLimit limit;
  cudaStream_t stream;
  unsigned int flags;
  cudaSharedMemConfig config;

  cudaDeviceSetLimit(limit, 0);
  cudaStreamAttachMemAsync(stream, nullptr);
  cudaStreamQuery(stream);
  cudaDeviceSetSharedMemConfig(config);
  cudaDeviceSetCacheConfig(cudaFuncCachePreferNone);
  cudaSetDeviceFlags(flags);
  cuInit(0);
  float *aptr;
  cudaStreamAttrValue stream_attribute = {};
  stream_attribute.accessPolicyWindow.base_ptr = aptr;
  stream_attribute.accessPolicyWindow.num_bytes = 8 * 8 * 8;
  stream_attribute.accessPolicyWindow.hitRatio = 1.0f;
  stream_attribute.accessPolicyWindow.hitProp = cudaAccessPropertyPersisting;
  stream_attribute.accessPolicyWindow.missProp = cudaAccessPropertyStreaming;
  cudaStreamSetAttribute(stream, cudaStreamAttributeAccessPolicyWindow, &stream_attribute);
  cudaStreamGetAttribute(stream, cudaStreamAttributeAccessPolicyWindow, &stream_attribute);
  cudaCtxResetPersistingL2Cache();
  cuCtxResetPersistingL2Cache();
  cudaFuncSetAttribute(kernel, cudaFuncAttributeMaxDynamicSharedMemorySize, 8 * 8 * 8);
  cudaFuncSetAttribute(kernel, cudaFuncAttributePreferredSharedMemoryCarveout, 8 * 8 * 8);
}
```

results in the following migrated SYCL code:

```
void bar(int *a) {
  /*
  DPCT1026:0: The call to __ldg was removed because there is no corresponding
  API in SYCL.
  */
  *a;
}

void kernel() {
  return;
}

void foo() {
  cudaLimit limit;
```

```
dpct::queue_ptr stream;
unsigned int flags;
int config;

/*
DPCT1026:1: The call to cudaDeviceSetLimit was removed because SYCL currently
does not support setting resource limits on devices.
*/
/*
DPCT1026:2: The call to cudaStreamAttachMemAsync was removed because SYCL
currently does not support associating USM with a specific queue.
*/
/*
DPCT1026:3: The call to cudaStreamQuery was removed because SYCL currently
does not support query operations on queues.
*/
/*
DPCT1026:4: The call to cudaDeviceSetSharedMemConfig was removed because SYCL
currently does not support configuring shared memory on devices.
*/
/*
DPCT1026:5: The call to cudaDeviceSetCacheConfig was removed because SYCL
currently does not support setting cache config on devices.
*/
/*
DPCT1026:6: The call to cudaSetDeviceFlags was removed because SYCL currently
does not support setting flags for devices.
*/
/*
DPCT1026:7: The call to cuInit was removed because this call is redundant in
SYCL.
*/
float *aptr;
/*
DPCT1007:8: Migration of cudaStreamAttrValue is not supported.
*/
cudaStreamAttrValue stream_attribute = {};
/*
DPCT1026:9: The call to cudaStreamSetAttribute was removed because SYCL currently
does not support setting cache config on devices.
*/
/*
DPCT1026:10: The call to cudaStreamGetAttribute was removed because SYCL currently
does not support setting cache config on devices.
*/
/*
DPCT1026:11: The call to cudaCtxResetPersistingL2Cache was removed because SYCL
currently does not support setting cache config on devices.
*/
/*
DPCT1026:12: The call to cuCtxResetPersistingL2Cache was removed because SYCL
currently does not support setting cache config on devices.
*/
/*
DPCT1026:13: The call to cudaFuncSetAttribute was removed because SYCL currently
does not support corresponding setting.
*/
/*
```

```
  DPCT1026:14: The call to cudaFuncSetAttribute was removed because SYCL currently
  does not support corresponding setting.
  */
}
```

which is rewritten to:

```
void bar(int *a) {
  *a;
}

void foo() {
  cudaLimit limit;
  dpct::queue_ptr stream;
  unsigned int flags;
  size_t count;
  int config;
  float *aptr;

}
```

DPCT1027

Message

The call to <*API name*> was replaced with 0 because <*reason*>.

Detailed Help

API calls from the original application, which do not have functionally compatible SYCL* API calls are replaced with 0 if Intel® DPC++ Compatibility Tool determines that this call removal should not affect the program logic and at the same time the return value of that call is used for error handling.

Possible reasons for replacement:

- SYCL currently does not support setting resource limits on devices.
- SYCL currently does not support associating USM with a specific queue.
- SYCL currently does not support query operations on queues.
- SYCL currently does not support capture operations on queues.
- SYCL currently does not support configuring shared memory on devices.
- SYCL currently does not support setting cache config on devices.
- SYCL currently does not support registering of existing host memory for use by device. Use USM to allocate memory for use by host and device.
- SYCL currently does not support setting flags for devices.
- SYCL currently does not support memory access across peer devices.
- There is no corresponding API in SYCL.
- The call is redundant in SYCL.

Suggestions to Fix

Verify the code correctness.

For example, this original CUDA* code:

```
__global__ void kernel() {
  return;
}

void foo() {
  int err = 0;
  cudaLimit limit;
  cudaStream_t stream;
  unsigned int flags;
  cudaSharedMemConfig config;

  err = cudaDeviceSetLimit(limit, 0);
  err = cudaStreamAttachMemAsync(stream, nullptr);
  err = cudaStreamQuery(stream);
  err = cudaStreamBeginCapture(stream, cudaStreamCaptureModeGlobal);
  err = cudaDeviceSetSharedMemConfig(config);
  err = cudaDeviceSetCacheConfig(cudaFuncCachePreferNone);
  err = cuMemHostRegister(ptr, count, flags);
  err = cudaSetDeviceFlags(flags);
  err = cudaDeviceEnablePeerAccess(peer_device, flags);
  err = cuInit(0);
  float *aptr;
  cudaStreamAttrValue stream_attribute = {};
  stream_attribute.accessPolicyWindow.base_ptr = aptr;
  stream_attribute.accessPolicyWindow.num_bytes = 8 * 8 * 8;
  stream_attribute.accessPolicyWindow.hitRatio = 1.0f;
  stream_attribute.accessPolicyWindow.hitProp = cudaAccessPropertyPersisting;
  stream_attribute.accessPolicyWindow.missProp = cudaAccessPropertyStreaming;
  err = cudaStreamSetAttribute(stream, cudaStreamAttributeAccessPolicyWindow, &stream_attribute);
  err = cudaStreamGetAttribute(stream, cudaStreamAttributeAccessPolicyWindow, &stream_attribute);
  err = cudaCtxResetPersistingL2Cache();
  err = cuCtxResetPersistingL2Cache();
  err = cudaFuncSetAttribute(kernel, cudaFuncAttributeMaxDynamicSharedMemorySize, 8 * 8 * 8);
}
```

results in the following migrated SYCL code:

```
void kernel() {
  return;
}

void foo() try {
  int err = 0;
  cudaLimit limit;
  dpct::queue_ptr stream;
  unsigned int flags;
  int config;

  /*
  DPCT1027:0: The call to cudaDeviceSetLimit was replaced with 0 because SYCL
  currently does not support setting resource limits on devices.
  */
  err = 0;
  /*
  DPCT1027:1: The call to cudaStreamAttachMemAsync was replaced with 0 because
```

```
SYCL currently does not support associating USM with a specific queue.
*/
err = 0;
/*
DPCT1027:2: The call to cudaStreamQuery was replaced with 0 because SYCL
currently does not support query operations on queues.
*/
err = 0;
/*
DPCT1027:3: The call to cudaDeviceSetSharedMemConfig was replaced with 0
because SYCL currently does not support configuring shared memory on devices.
*/
err = 0;
/*
DPCT1027:4: The call to cudaDeviceSetCacheConfig was replaced with 0 because
SYCL currently does not support setting cache config on devices.
*/
err = 0;
/*
DPCT1027:5: The call to cudaSetDeviceFlags was replaced with 0 because SYCL
currently does not support setting flags for devices.
*/
err = 0;
/*
DPCT1027:6: The call to cuInit was replaced with 0 because this call is
redundant in SYCL.
*/
err = 0;
float *aptr;
/*
DPCT1007:7: Migration of cudaStreamAttributeValue is not supported.
*/
cudaStreamAttrValue stream_attribute = {};
/*
DPCT1027:8: The call to cudaStreamSetAttribute was removed because SYCL currently
does not support setting cache config on devices.
*/
err = 0;
/*
DPCT1027:9: The call to cudaStreamGetAttribute was removed because SYCL currently
does not support setting cache config on devices.
*/
err = 0;
/*
DPCT1027:10: The call to cudaCtxResetPersistingL2Cache was removed because SYCL
currently does not support setting cache config on devices.
*/
err = 0;
/*
DPCT1027:11: The call to cuCtxResetPersistingL2Cache was removed because SYCL
currently does not support setting cache config on devices.
*/
err = 0;
/*
DPCT1027:12: The call to cudaFuncSetAttribute was removed because SYCL currently
does not support corresponding setting.
*/
err = 0;
```

```
  /*
  DPCT1027:13: The call to cudaFuncSetAttribute was removed because SYCL currently
  does not support corresponding setting.
  */
  err = 0;
}
catch (sycl::exception const &exc) {
  std::cerr << exc.what() << "Exception caught at file:" << __FILE__
            << ", line:" << __LINE__ << std::endl;
  std::exit(1);
}
```

which is rewritten to:

```
void foo() {
  int err = 0;
  cudaLimit limit;
  dpct::queue_ptr stream;
  unsigned int flags;
  int config;
  float *aptr;
}
```

DPCT1028

Message

The <*API name*> was not migrated because <*reason*>.

Detailed Help

Intel® DPC++ Compatibility Tool was not able to recognize the parameter value used as a CUDA* device attribute.

Suggestions to Fix

Redesign the code logic to remove use of the device attribute.

For example, this original CUDA code:

```
void foo(int *result, CUdevice device) {
  cuDeviceGetAttribute(result, CU_DEVICE_ATTRIBUTE_GPU_OVERLAP, device);
}
```

results in the following migrated SYCL* code:

```
void foo(int *result, int device) {
  /*
  DPCT1028:0: The cuDeviceGetAttribute was not migrated because parameter
  CU_DEVICE_ATTRIBUTE_GPU_OVERLAP is unsupported.
```

```
  */
  cuDeviceGetAttribute(result, CU_DEVICE_ATTRIBUTE_GPU_OVERLAP, device);
}
```

which is rewritten to:

```
void foo(int *result, int device) {
  // Assume the HW supports copy memory and execute a kernel concurrently
  *result = 1;
}
```

DPCT1029

Message

SYCL currently does not support getting device resource limits. The output parameter(s) are set to 0.

Detailed Help

SYCL* currently does not support getting device resource limits. The output parameter(s) are set to 0.

Visit the Migrating to SYCL forum for additional help.

Suggestions to Fix

Review the logic and adjust it manually.

For example, this original CUDA* code:

```
void foo(size_t *value) {
  cudaDeviceGetLimit(value, cudaLimitPrintfFifoSize);
}
```

results in the following migrated SYCL code:

```
void foo(size_t *value) {
  /*
  DPCT1029:0: SYCL currently does not support getting device resource limits.
  The output parameter(s) are set to 0.
  */
  *value = 0;
}
```

which is rewritten to:

```
void foo(size_t *value) {
  *value = totalBufferSize /* the first argument of the constructor of sycl::stream */;
}
```

DPCT1030

Message

SYCL currently does not support inter-process communication (IPC) operations. You may need to rewrite the code.

Detailed Help

IPC operations are currently not supported by SYCL*.

Visit the Migrating to SYCL forum for additional help.

Suggestions to Fix

Rewrite the code manually.

DPCT1031

Message

Memory access across peer devices is an implementation-specific feature which may not be supported by some SYCL backends and compilers. The output parameter(s) are set to 0. You can migrate the code with peer access extension if you do not specify -no-dpcpp-extensions=peer_access.

Detailed Help

Memory access across peer devices is an implementation-specific feature which may not be supported by some SYCL* backends and compilers. The output parameter(s) are set to 0. You can migrate the code with peer access extension by default without specifying -no-dpcpp-extensions=peer_access.

Visit the Migrating to SYCL forum for additional help.

Suggestions to Fix

Migrate the code with peer access extension by not specifying -no-dpcpp-extensions=peer_access.

DPCT1032

Message

A different random number generator is used. You may need to adjust the code.

Detailed Help

Intel® oneAPI Math Kernel Library (oneMKL) RNG currently does not support the following engine types:

- XORWOW generator
- Sobol generator of 64-bit sequences
- Scrambled Sobol generator

The engine types are migrated to the following supported generators, respectively:

- MCG59 generator
- Basic Sobol generator
- Basic Sobol generator

Suggestions to Fix

You may need to adjust the code.

For example, this original CUDA* code:

```
__global__ void kernel(curandStateXORWOW_t rng) {
  ...
}
```

results in the following migrated SYCL* code:

```
/*
DPCT1032:0: A different random number generator is used. You may need to adjust
the code.
*/
void kernel(
    dpct::rng::device::rng_generator<oneapi::mkl::rng::device::mcg59<1>> rng) {
  ...
}
```

which is rewritten to:

```
void kernel(
    dpct::rng::device::rng_generator<oneapi::mkl::rng::device::mcg59<1>> rng) {
  ...
}
```

DPCT1033

Message

Migrated code uses a basic Sobol generator. Initialize `oneapi::mkl::rng::sobol` generator with user-defined direction numbers to use it as Scrambled Sobol generator.

Detailed Help

Intel® oneAPI Math Kernel Library (oneMKL) RNG currently does not support Scrambled Sobol generator by default.

Migrated code uses a basic Sobol generator.

Suggestions to Fix

Set user-defined direction numbers to the basic Sobol generator and use it as Scrambled Sobol generator.

See the Random Number Generators topic for more information.

For example, this original CUDA* code:

```
void foo() {
  ...
  curandGenerator_t rng;
  curandCreateGenerator(&rng, CURAND_RNG_QUASI_SCRAMBLED_SOBOL32);
  curandSetQuasiRandomGeneratorDimensions(rng, 1234);
}
```

results in the following migrated SYCL* code:

```
void foo() {
  ...
  dpct::rng::host_rng_ptr rng;
  /*
  DPCT1033:0: Migrated code uses a basic Sobol generator. Initialize
  oneapi::mkl::rng::sobol generator with user-defined direction numbers to use
  it as Scrambled Sobol generator.
  */
  rng = dpct::rng::create_host_rng(dpct::rng::random_engine_type::sobol);
  rng->set_dimensions(1234);
}
```

which is rewritten to:

```
void foo() {
  ...
  dpct::rng::host_rng_ptr rng;
  rng = dpct::rng::create_host_rng(dpct::rng::random_engine_type::sobol);
  rng->set_direction_numbers(direction_numbers /*Use user-defined direction numbers*/);
}
```

DPCT1034

Message

Migrated API does not return an error code. 0 is returned in the lambda. You may need to rewrite this code.

Detailed Help

Typically, this happens because the API call in the original application returns an error code and then it is consumed by the program logic.

SYCL* uses exceptions to report errors and does not return the error code.

Intel® DPC++ Compatibility Tool inserts a `return 0;` statement at the end of the lambda expression, if the return code is expected by the program logic and the new API does not return it. Review all such places in the code.

Similar to DPCT1003.

Suggestions to Fix

If in a DPC++ application you:

- Do not need the code that consumes the error code, remove the code and the `return 0;` statement.
- Need the code that consumes the error code, try to replace it with an exception handling code and use your logic in an exception handler.

Similar to DPCT1003.

For example, this original CUDA* code:

```
void foo() {
  CUdevice device;
  CUresult status = cuDeviceComputeCapability(&result0, &result1, device);
}
```

results in the following migrated SYCL code:

```
void foo() {
  /*
  DPCT1034:0: Migrated API does not return an error code. 0 is returned in the
  lambda. You may need to rewrite this code.
  */
  int status = [&]() {
    result0 = dpct::dev_mgr::instance().get_device(device).get_major_version();
    result1 = dpct::dev_mgr::instance().get_device(device).get_minor_version();
    return 0;
  }();
}
```

which is rewritten to:

```
void foo() {
  try {
    result0 = dpct::dev_mgr::instance().get_device(device).get_major_version();
    result1 = dpct::dev_mgr::instance().get_device(device).get_minor_version();
  } catch (...) {
    ...
  }
}
```

DPCT1035

Message

All SYCL devices can be used by the host to submit tasks. You may need to adjust this code.

Detailed Help

In SYCL*, the host can freely submit tasks to all SYCL devices. There's no need to check for device compute mode.

Suggestions to Fix

Review the code and adjust accordingly.

For example, this original CUDA* code:

```
void foo(cudaDeviceProp deviceProp) {
  if (deviceProp.computeMode == cudaComputeModeDefault) {
    ... // code path 1
  } else if (deviceProp.computeMode == cudaComputeModeExclusive) {
    ... // code path 2
  } else if (deviceProp.computeMode == cudaComputeModeProhibited) {
    ... // code path 3
  }
}
```

results in the following migrated SYCL code:

```
void foo(dpct::device_info deviceProp) {
  /*
  DPCT1035:0: All SYCL devices can be used by the host to submit tasks. You may
  need to adjust this code.
  */
  if (true) {
    ... // code path 1
    /*
    DPCT1035:1: All SYCL devices can be used by the host to submit tasks. You may
    need to adjust this code.
    */
  } else if (false) {
    ... // code path 2
    /*
    DPCT1035:2: All SYCL devices can be used by the host to submit tasks. You may
    need to adjust this code.
    */
  } else if (false) {
    ... // code path 3
  }
}
```

which is rewritten to:

```
void foo(dpct::device_info deviceProp) {
  ... // code path 1
}
```

DPCT1036

NOTE This diagnostic message is no longer generated by Intel® DPC++ Compatibility Tool 2023.2 or later.

Message

The type <*type name*> was not migrated because <*reason*>.

Detailed Help

Intel® DPC++ Compatibility Tool was unable to analyze and migrate the API call, so the <*type name*> type was not migrated.

See DPCT1028 for more details.

Suggestions to Fix

Rewrite this code manually by using a supported random number generator.

See the Random Number Generators topic for more information.

DPCT1037

Message

Rewrite this code using <*library name*> with SYCL.

Detailed Help

Intel® DPC++ Compatibility Tool does not support the migration of some library API calls.

Suggestions to Fix

Rewrite this code manually by using the Intel® oneAPI Deep Neural Network Library (oneDNN) or the Intel® oneAPI Collective Communications Library (oneCCL).

DPCT1038

Message

When the kernel function name is used as a macro argument, the migration result may be incorrect. You need to verify the definition of the macro.

Detailed Help

When a kernel function call is used in a macro definition and the function name is passed as a macro argument, the tool cannot determine the type of function parameters. This may lead to incorrectly generated DPC++ code.

Suggestions to Fix

Review the kernel call inside the macro definition and adjust it manually, if needed.

For example, this original CUDA* code:

```
__global__ void kernel(int c, int d) {}

void foo() {
#define KERNEL_LAUNCH(NAME, a, b, c, d) NAME<<<a, b, 0>>>(c, d);
  KERNEL_LAUNCH(kernel, 3, 2, 1, 0)
}
```

results in the following migrated SYCL* code:

```
void kernel(int c, int d) {}

void foo() {
#define KERNEL_LAUNCH(NAME, a, b, c, d)                                        \
  dpct::get_default_queue().submit([&](sycl::handler &cgh) {                   \
    auto c_ct0 = c;                                                            \
    auto d_ct1 = d;                                                            \
                                                                               \
    cgh.parallel_for(                                                          \
        sycl::nd_range<3>(sycl::range<3>(1, 1, a) * sycl::range<3>(1, 1, b),   \
                          sycl::range<3>(1, 1, b)),                            \
        [=](sycl::nd_item<3> item_ct1) { kernel(c_ct0, d_ct1); });             \
  });
  /*
  DPCT1038:0: When the kernel function name is used as a macro argument, the
  migration result may be incorrect. You need to verify the definition of the
  macro.
  */
  KERNEL_LAUNCH(kernel, 3, 2, 1, 0)
}
```

which is rewritten to:

```
void kernel(int c, int d) {}

void foo() {
#define KERNEL_LAUNCH(NAME, a, b, c, d)                                        \
  dpct::get_default_queue().submit([&](sycl::handler &cgh) {                   \
```

```
    auto c_ct0 = c;                                                          \
    auto d_ct1 = d;                                                          \
                                                                             \
    cgh.parallel_for(                                                        \
        sycl::nd_range<3>(sycl::range<3>(1, 1, a) * sycl::range<3>(1, 1, b),   \
                          sycl::range<3>(1, 1, b)),                          \
        [=](sycl::nd_item<3> item_ct1) { NAME(c_ct0, d_ct1); });             \
  });
  KERNEL_LAUNCH(kernel, 3, 2, 1, 0)
}
```

DPCT1039

Message

The generated code assumes that <*parameter name*> points to the global memory address space. If it points to a local memory address space, replace <*function name*> with <*function name*>.

Detailed Help

Intel® DPC++ Compatibility Tool deduces whether the first parameter of an atomic function points to a global memory address space or a local memory space, using the last assignment's RHS value of the first parameter of the atomic function. If the last assignment is in an `if/while/do/for` statement or the first argument of the atomic function depends on a function or template argument, the deduction result may be incorrect. You need to verify the generated code to determine if the first parameter of the atomic function actually points to the local memory address space. If it does, then replace the atomic function name with an atomic function name that includes the template parameters, as pointed to in the warning message.

Suggestions to Fix

If the first parameter of an atomic function points to a local memory address space, replace the atomic function name with an atomic function name that includes the template parameters.

For example, this original CUDA* code:

```
__device__ void d(int* s) {
  atomicAdd(s, 1);
}

__global__ void k() {
  __shared__ int shared;
  d(&shared);
}
```

results in the following migrated SYCL* code:

```
void d(int* s) {
  /*
  DPCT1039:0: The generated code assumes that "s" points to the global memory
  address space. If it points to a local memory address space, replace
```

```
  "atomic_fetch_add" with
  "atomic_fetch_add<sycl::access::address_space::local_space>".
  */
  dpct::atomic_fetch_add(s, 1);
}

void k(int &shared) {
  d(&shared);
}
```

which is rewritten to:

```
template<sycl::access::address_space as>
void d(int* s) {
  dpct::atomic_fetch_add<as>(s, 1);
}

void k(int &shared) {
  d<sycl::access::address_space::local_space>(&shared);
}
```

DPCT1040

Message

Use `sycl::stream` instead of `printf` if your code is used on the device.

Detailed Help

If the `printf` statement is used on the host and the device in your original code, it does not change. To create output in DPC++, `sycl::stream` must be used on the device and `printf` must be used on the host.

Suggestions to Fix

- If the `printf` statement is only used from the host, do not change your code.
- If the `printf` statement is only used from the device, use `sycl::stream` instead of `printf`.

For example, this original CUDA* code:

```
__host__ __device__ void hd() {
  printf("Hello!\n");
}

__global__ void k() {
  hd();
}

void foo() {
```

```
  hd();
  k<<<1, 1>>>();
}
```

results in the following migrated SYCL* code:

```
void hd() {
  /*
  DPCT1040:0: Use sycl::stream instead of printf if your code is used on the
  device.
  */
  printf("Hello!\n");
}

void k() {
  hd();
}

void foo() {
  hd();
  dpct::get_default_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
        k();
      });
}
```

which is rewritten to:

```
void hd_host() {
  printf("Hello!\n");
}

void hd_device(const sycl::stream &stream) {
  stream << "Hello!\n";
}

void k(const sycl::stream &stream) {
  hd_device(stream);
}

void foo() {
  hd_host();
  dpct::get_default_queue().submit([&](sycl::handler &cgh) {
    sycl::stream stream(64 * 1024 /*totalBufferSize*/, 80 /*workItemBufferSize*/, cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          k(stream);
        });
  });
}
```

DPCT1041

Message

SYCL uses exceptions to report errors, it does not use error codes. 0 is used instead of an error code in *<statement name>* statement. You may need to rewrite this code.

Detailed Help

SYCL* uses exceptions to report errors, it does not use error codes. The original code tries to get an error code, but SYCL does not require this functionality. Therefore, 0 is used instead of an error code in the following statements:

- if
- while
- do
- for
- switch
- return
- function-like macro

Suggestions to Fix

You may need to rewrite this code.

For example, this original CUDA* code:

```
void foo() {
  ...
  int N;
  cublasHandle_t handle;
  float alpha, beta;
  float *a, *b, *c;
  if (int status = cublasSgemm(handle, CUBLAS_OP_N, CUBLAS_OP_N, N, N, N, &alpha,
                               a, N, b, N, &beta, c, N)) {
    printf("cublasSgemm failed\n");
  }
  ...
}
```

results in the following migrated SYCL code:

```
void foo() try {
  ...
  int N;
  dpct::queue_ptr handle;
  float alpha, beta;
  float *a, *b, *c;
  {
    auto a_buf_ct1 = dpct::get_buffer<float>(a);
    auto b_buf_ct2 = dpct::get_buffer<float>(b);
    auto c_buf_ct3 = dpct::get_buffer<float>(c);
    oneapi::mkl::blas::column_major::gemm(
        *handle, oneapi::mkl::transpose::nontrans,
        oneapi::mkl::transpose::nontrans, N, N, N, alpha, a_buf_ct1, N,
```

```
        b_buf_ct2, N, beta, c_buf_ct3, N);
  }
  /*
  DPCT1041:0: SYCL uses exceptions to report errors, it does not use error
  codes. 0 is used instead of an error code in an if statement. You may need to
  rewrite this code.
  */
  if (int status = 0) {
    printf("cublasSgemm failed\n");
  }
  ...
}
catch (sycl::exception const &exc) {
  std::cerr << exc.what() << "Exception caught at file:" << __FILE__
            << ", line:" << __LINE__ << std::endl;
  std::exit(1);
}
```

which is rewritten to:

```
void foo() try {
  ...
  int N;
  dpct::queue_ptr handle;
  float alpha, beta;
  float *a, *b, *c;
  {
    auto a_buf_ct1 = dpct::get_buffer<float>(a);
    auto b_buf_ct2 = dpct::get_buffer<float>(b);
    auto c_buf_ct3 = dpct::get_buffer<float>(c);
    oneapi::mkl::blas::column_major::gemm(
        *handle, oneapi::mkl::transpose::nontrans,
        oneapi::mkl::transpose::nontrans, N, N, N, alpha, a_buf_ct1, N,
        b_buf_ct2, N, beta, c_buf_ct3, N);
  }
  ...
}
catch (sycl::exception const &exc) {
  std::cerr << exc.what() << "Exception caught at file:" << __FILE__
            << ", line:" << __LINE__ << std::endl;
  std::exit(1);
}
```

DPCT1042

Message

The size of the arguments passed to the SYCL kernel exceeds the minimum size limit (1024) for a non-custom SYCL device. You can get the hardware argument size limit by querying `info::device::max_parameter_size`. You may need to rewrite this code if the size of the arguments exceeds the hardware limit.

Detailed Help

The size of the arguments passed to the SYCL* kernel for non-custom SYCL device has a limit (see SYCL 2020 standard, 4.6.4.2 Device information descriptors).

In cases where this warning occurs, you need to adjust the code manually to decrease the number of accessors or other arguments that are captured by the SYCL kernel lambda.

The example in the next section shows how you can remove one accessor by merging two buffers with the same type.

Suggestions to Fix

Review the code and adjust it.

For example, this original CUDA* code:

```
#define ARRAY_SIZE 2

__constant__ int device0[ARRAY_SIZE];
__constant__ int device1[ARRAY_SIZE];
__constant__ int device2[ARRAY_SIZE];
...
__constant__ int device30[ARRAY_SIZE];
__constant__ int device31[ARRAY_SIZE];

// kernel function declaration
__global__ void kernel(int *out) {
  int i = blockDim.x * blockIdx.x + threadIdx.x;
  out[i] = device0[i] + device1[i] + device2[i] + ... +
           device30[i] + device31[i];
}

void test_function(int *out) {
  kernel<<<1, 1>>>(out);
}
```

results in the following migrated SYCL code:

```
#define ARRAY_SIZE 2

static dpct::constant_memory<int, 1> device0(ARRAY_SIZE);
static dpct::constant_memory<int, 1> device1(ARRAY_SIZE);
static dpct::constant_memory<int, 1> device2(ARRAY_SIZE);
...
static dpct::constant_memory<int, 1> device30(ARRAY_SIZE);
static dpct::constant_memory<int, 1> device31(ARRAY_SIZE);

// kernel function declaration
void kernel(int *out, const sycl::nd_item<3> &item_ct1, int const *device0,
            int const *device1, int const *device2, ..., int const *device30,
            int const *device31) {
  int i = item_ct1.get_local_range(2) * item_ct1.get_group(2) +
          item_ct1.get_local_id(2);
  out[i] = device0[i] + device1[i] + device2[i] +
           ... +
           device30[i] + device31[i];
}
```

```
void test_function(int *out) {
  device0.init();
  device1.init();
  device2.init();
  ...
  device30.init();
  device31.init();

  /*
  DPCT1042:0: The size of the arguments passed to the SYCL kernel exceeds the
  minimum size limit (1024) for a non-custom SYCL device. You can get the
  hardware argument size limit by querying info::device::max_parameter_size. You
  may need to rewrite this code if the size of the arguments exceeds the
  hardware limit.
  */
  dpct::get_out_of_order_queue().submit([&](sycl::handler &cgh) {
    auto device0_acc_ct1 = device0.get_access(cgh);
    auto device1_acc_ct1 = device1.get_access(cgh);
    auto device2_acc_ct1 = device2.get_access(cgh);
    ...
    auto device30_acc_ct1 = device30.get_access(cgh);
    auto device31_acc_ct1 = device31.get_access(cgh);
    dpct::access_wrapper<int *> out_acc_ct0(out, cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel(out_acc_ct0.get_raw_pointer(), item_ct1,
                 device0_acc_ct1.get_pointer(), device1_acc_ct1.get_pointer(),
                 device2_acc_ct1.get_pointer(), ...,
                 device30_acc_ct1.get_pointer(), device31_acc_ct1.get_pointer());
        });
  });
}
```

which is rewritten to:

```
#define ARRAY_SIZE 2

static dpct::constant_memory<int, 1> device0(ARRAY_SIZE * 32);

// kernel function declaration
void kernel(int *out, const sycl::nd_item<3> &item_ct1, int const *device0) {
  int i = item_ct1.get_local_range(2) * item_ct1.get_group(2) +
          item_ct1.get_local_id(2);
  for (int j = 0; j < 32; j++) {
    out[i] += device0[ARRAY_SIZE * j + i];
  }
}

void test_function(int *out) {
  device0.init();

  dpct::get_out_of_order_queue().submit([&](sycl::handler &cgh) {
    auto device0_acc_ct1 = device0.get_access(cgh);
    dpct::access_wrapper<int *> out_acc_ct0(out, cgh);

    cgh.parallel_for(
```

```
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel(out_acc_ct0.get_raw_pointer(), item_ct1,
                 device0_acc_ct1.get_pointer());
        });
  });
}
```

DPCT1043

Message

The version-related API is different in SYCL. An initial code was generated, but you need to adjust it.

Detailed Help

The generated code uses `sycl::info::device::version`, which provides different information and uses a different type than the original code.

Suggestions to Fix

Adjust the generated code.

For example, this original CUDA* code:

```
void foo() {
  int version;
  cudaDriverGetVersion(&version);
  if (version >= 12000) {
    code path 1
  } else {
    code path 2
  }
}
```

results in the following migrated SYCL* code:

```
void foo() {
  int version;
  /*
  DPCT1043:0: The version-related API is different in SYCL. An initial code was
  generated, but you need to adjust it.
  */
  version = dpct::get_current_device().get_major_version();
  if (version >= 12000) {
    ...
  } else {
    ...
  }
}
```

which is rewritten to:

```
void foo() {
  code path 1
}
```

DPCT1044

Message

<BaseClass1> was removed because *<BaseClass2>* has been deprecated in C++11. You may need to remove references to typedefs from *<BaseClass1>* in the class definition.

Detailed Help

<BaseClass#> is either `unary_function` or `binary_function`. The `std::` equivalents for these classes are deprecated, and the use of these base classes is removed.

Suggestions to Fix

If any of the typedef identifiers are referenced in the class definition, they should be replaced with the original template arguments.

For example, this original CUDA* code:

```
class C : thrust::unary_function<int, float> {
  argument_type arg_data;
  result_type result_data;
};
```

results in the following migrated SYCL* code:

```
/*
DPCT1044:0: thrust::unary_function was removed because std::unary_function has
been deprecated in C++11. You may need to remove references to typedefs from
thrust::unary_function in the class definition.
*/
class C {
  argument_type arg_data;
  result_type result_data;
};
```

which is rewritten to:

```
class C {
  argument_type arg_data;
  result_type result_data;
};
```

DPCT1045

Message

Migration is only supported for this API for the <matrix type> sparse matrix type. You may need to adjust the code.

Detailed Help

This warning appears if the matrix type in use is not supported or cannot be determined.

Suggestions to Fix

If the matrix type in used is:

- Supported by the routine: ignore this warning.
- Not supported by the routine: manually fix the code according to sparse-blas-routines.

For example, this original CUDA* code:

```
//     --          --
//     | 1   0   2 |
// A = | 0   3   4 |
//     | 0   0   5 |
//     --          --
// a_val = [1, 2, 3, 4, 5]
// a_row_ptr = [0, 2, 4, 5]
// a_col_ind = [0, 2, 1, 2, 2]
void test_cusparseTcsrmm(cusparseHandle_t handle, cusparseMatDescr_t descrA,
                         float alpha, float beta, float *b, float *c) {
  // malloc and set value
  float *a_val = my_malloc<float>();
  int *a_row_ptr = my_malloc<int>();
  int *a_col_ind = my_malloc<int>();
  set_value(a_val);
  set_value(a_row_ptr);
  set_value(a_col_ind);

  // calculate
  cusparseSetMatType(descrA, CUSPARSE_MATRIX_TYPE_SYMMETRIC);
  cusparseScsrmm(handle, CUSPARSE_OPERATION_NON_TRANSPOSE, 4, 2, 5, 9, &alpha,
                 descrA, a_val, a_row_ptr, a_col_ind, &beta, 5, b, c, 4);

  // synchronize and check value
  ...
  // free
  my_free(a_val);
  my_free(a_row_ptr);
  my_free(a_col_ind);
}
```

results in the following migrated SYCL* code:

```
//     --         --
//     | 1   0   2 |
// A = | 0   3   4 |
//     | 0   0   5 |
//     --         --
// a_val = [1, 2, 3, 4, 5]
// a_row_ptr = [0, 2, 4, 5]
// a_col_ind = [0, 2, 1, 2, 2]
void test_cusparseTcsrmm(sycl::queue *handle,
                         std::shared_ptr<dpct::sparse::matrix_info> descrA,
                         float alpha, float beta, float *b, float *c) {
  // malloc and set value
  float *a_val = my_malloc<float>();
  int *a_row_ptr = my_malloc<int>();
  int *a_col_ind = my_malloc<int>();
  set_value(a_val);
  set_value(a_row_ptr);
  set_value(a_col_ind);

  // calculate
  descrA->set_matrix_type(dpct::sparse::matrix_info::matrix_type::sy);
  /*
  DPCT1045:0: Migration is only supported for this API for the general sparse
  matrix type. You may need to adjust the code.
  */
  dpct::sparse::csrmm(*handle, oneapi::mkl::transpose::nontrans, 4, 2, 5,
                      &alpha, descrA, a_val, a_row_ptr, a_col_ind, &beta, 5, b,
                      c, 4);

  // synchronize and check value
  ...
  // free
  my_free(a_val);
  my_free(a_row_ptr);
  my_free(a_col_ind);
}
```

which is rewritten to:

```
//     --         --
//     | 1   0   2 |
// A = | 0   3   4 |
//     | 0   0   5 |
//     --         --
// a_val = [1, 2, 3, 4, 5]
// a_row_ptr = [0, 2, 4, 5]
// a_col_ind = [0, 2, 1, 2, 2]
//
// Original matrix A is a symmetric matrix. Only upper/lower data is used.
// To make it to be a general matrix, all data in the matrix need to be filled.
//         --         --
//         | 1   0   2 |
// new_A = | 0   3   4 |
//         | 2   4   5 |
//         --         --
// new_a_val = [1, 2, 3, 4, 2, 4, 5]
// new_a_row_ptr = [0, 2, 4, 7]
// new_a_col_ind = [0, 2, 1, 2, 0, 1, 2]
```

```
void test_cusparseTcsrmm(sycl::queue *handle,
                         std::shared_ptr<dpct::sparse::matrix_info> descrA,
                         float alpha, float beta, float *b, float *c) {
  // malloc and set value
  float *a_val = my_malloc<float>();
  int *a_row_ptr = my_malloc<int>();
  int *a_col_ind = my_malloc<int>();
  set_value(a_val);
  set_value(a_row_ptr);
  set_value(a_col_ind);

  float *new_a_val = my_malloc<float>();
  int *new_a_row_ptr = my_malloc<int>();
  int *new_a_col_ind = my_malloc<int>();

  // convert matrix data from symmetric format to general format
  std::tie(new_a_val, new_a_row_ptr, new_a_col_ind) =
      symmetric_to_general(a_val, a_row_ptr, a_col_ind);

  my_free(a_val);
  my_free(a_row_ptr);
  my_free(a_col_ind);

  // calculate
  descrA->set_matrix_type(dpct::sparse::matrix_info::matrix_type::ge);
  dpct::sparse::csrmm(*handle, oneapi::mkl::transpose::nontrans, 4, 2, 5,
                      &alpha, descrA, new_a_val, new_a_row_ptr,
                      new_a_col_ind, &beta, 5, b, c, 4);

  // synchronize and check value
  ...
  // free
  my_free(new_a_val);
  my_free(new_a_row_ptr);
  my_free(new_a_col_ind);
}
```

DPCT1046

NOTE This diagnostic message is no longer generated by Intel® DPC++ Compatibility Tool 2023.0 or later.

Message

The <*original API name*> was not migrated because <*reason*>. You need to adjust the code.

Detailed Help

Not all data type combinations are supported by `mkl::blas::gemm()`.

This may be due to one of the following reasons:

- Not all values of parameters could be evaluated in migration.
- The combination of matrix data type and scalar type is unsupported.

Use a supported data type to rewrite the code.

Suggestions to Fix

Please refer to the gemm topic of the Intel® oneAPI Math Kernel Library (oneMKL) - Data Parallel C++ Developer Reference for supported data types to fix the code manually.

For example, this original CUDA* code:

```
void foo(cublasHandle_t handle, float alpha, float beta, void *a, void *b,
         void *c, cudaDataType_t type) {
  cublasSgemmEx(handle, CUBLAS_OP_C, CUBLAS_OP_C, 2, 2, 2, &alpha, a, type, 2,
                b, type, 2, &beta, c, type, 2);
}
```

results in the following migrated SYCL* code:

```
void foo(dpct::queue_ptr handle, float alpha, float beta, void *a, void *b,
         void *c, dpct::library_data_t type) {
  /*
  DPCT1046:0: The cublasSgemmEx was not migrated because not all values of
  parameters could be evaluated in migration. You need to adjust the code.
  */
  cublasSgemmEx(handle, oneapi::mkl::transpose::conjtrans,
                oneapi::mkl::transpose::conjtrans, 2, 2, 2, &alpha, a, type, 2,
                b, type, 2, &beta, c, type, 2);
}
```

which is rewritten to:

```
void foo(dpct::queue_ptr handle, float alpha, float beta, void *a, void *b,
         void *c, dpct::library_data_t type) {
  switch (type) {
    case dpct::library_data_t::real_bfloat16: {
      oneapi::mkl::blas::column_major::gemm(
          *handle, oneapi::mkl::transpose::conjtrans,
          oneapi::mkl::transpose::conjtrans, 2, 2, 2, alpha,
          (oneapi::mkl::bfloat16 *)a, 2, (oneapi::mkl::bfloat16 *)b, 2, beta,
          (oneapi::mkl::bfloat16 *)c, 2);
      break;
    }
    case dpct::library_data_t::real_half: {
      oneapi::mkl::blas::column_major::gemm(
          *handle, oneapi::mkl::transpose::conjtrans,
          oneapi::mkl::transpose::conjtrans, 2, 2, 2, alpha, (sycl::half *)a, 2,
          (sycl::half *)b, 2, beta, (sycl::half *)c, 2);
      break;
    }
    case dpct::library_data_t::real_float: {
      oneapi::mkl::blas::column_major::gemm(
          *handle, oneapi::mkl::transpose::conjtrans,
          oneapi::mkl::transpose::conjtrans, 2, 2, 2, alpha, (float *)a, 2,
          (float *)b, 2, beta, (float *)c, 2);
      break;
    }
```

```
    default:
      throw std::runtime_error("the data type is unsupported");
  }
}
```

DPCT1047

Message

The meaning of <*parameter name*> in the <*API name*> is different from the <*API name*>. You may need to check the migrated code.

Detailed Help

In cuBLAS* and cuSolver* getrf API, the LU factorization is done as `P*A=L*U`; in Intel® oneAPI Math Kernel Library API, it is done as `A=P*L*U`. The result of the matrix P may be different.

Suggestions to Fix

If the matrix P is only used in library API, ignore this warning. If P is used other ways, you may need to adjust the value of P.

For example, this original CUDA* code:

```
void foo(cublasHandle_t handle, float **a_array, float **b_array, int *p_array,
         int *info_array) {
  cublasSgetrfBatched(handle, 2, a_array, 2, p_array, info_array, 1);
  cublasSgetrsBatched(handle, CUBLAS_OP_N, 2, 2, a_array, 2, p_array, b_array,
                      2, info_array, 1);
}
```

results in the following migrated SYCL* code:

```
void foo(dpct::queue_ptr handle, float **a_array, float **b_array, int *p_array,
         int *info_array) {
  /*
  DPCT1047:0: The meaning of p_array in the dpct::getrf_batch_wrapper is
  different from the cublasSgetrfBatched. You may need to check the migrated
  code.
  */
  dpct::getrf_batch_wrapper(*handle, 2, a_array, 2, p_array, info_array, 1);
  dpct::getrs_batch_wrapper(*handle, oneapi::mkl::transpose::nontrans, 2, 2,
                            const_cast<float const **>(a_array), 2, p_array,
                            b_array, 2, info_array, 1);
}
```

which is rewritten to:

```
void foo(dpct::queue_ptr handle, float **a_array, float **b_array, int *p_array,
         int *info_array) {
  // cublas/cusolver API getrs need the output of cublas/cusolver getrf as input.
```

```
  // MKL/dpct helper API getrs need the output of MKL/dpct helper getrf as input.
  // In this case, matrix P is only used as a temporary data between library API
  // invocations, so the warning can be ignored.
  dpct::getrf_batch_wrapper(*handle, 2, a_array, 2, p_array, info_array, 1);
  dpct::getrs_batch_wrapper(*handle, oneapi::mkl::transpose::nontrans, 2, 2,
                            const_cast<float const **>(a_array), 2, p_array,
                            b_array, 2, info_array, 1);
}
```

DPCT1048

Message

The original value <*macro name*> is not meaningful in the migrated code and was removed or replaced with 0. You may need to check the migrated code.

Detailed Help

When API (calls, flags, etc.) are not meaningful in DPC++, they may be removed or replaced with 0, depending on how they are used.

Suggestions to Fix

Review the code and adjust it.

For example, this original CUDA* code:

```
void foo() {
  double2 *h_A;
  cudaHostAlloc(&h_A, sizeof(double2), cudaHostAllocDefault);
}
```

results in the following migrated SYCL* code:

```
void foo() {
  sycl::double2 *h_A;
  /*
  DPCT1048:0: The original value cudaHostAllocDefault is not meaningful in the
  migrated code and was removed or replaced with 0. You may need to check the
  migrated code.
  */
  h_A = sycl::malloc_host<sycl::double2>(1, dpct::get_default_queue());
}
```

which is rewritten to:

```
void foo() {
  sycl::double2 *h_A;
  h_A = sycl::malloc_host<sycl::double2>(1, dpct::get_default_queue());
}
```

DPCT1049

Message

The work-group size passed to the SYCL kernel may exceed the limit. To get the device limit, query `info::device::max_work_group_size`. Adjust the work-group size if needed.

Detailed Help

The work-group size passed to the SYCL* kernel for SYCL device has a limit (see SYCL 2020 standard, 4.6.4.2 Device information descriptors).

This warning appears if dimensions of the local range could not all be evaluated, or if the product of the dimensions of the local range is more than 256.

Suggestions to Fix

Query `info::device::max_work_group_size` to define the work-group size limit for the device you use. If the work-group size used in the code is below the limit, you can ignore this warning. Otherwise, you need to decrease the work-group size.

For example, this original CUDA* code:

```
__global__ void k() {}

void foo() {
  k<<<1, 2048>>>();
}
```

results in the following migrated SYCL code:

```
void k() {}

void foo() {
  /*
  DPCT1049:0: The work-group size passed to the SYCL kernel may exceed the
  limit. To get the device limit, query info::device::max_work_group_size.
  Adjust the work-group size if needed.
  */
  dpct::get_default_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 2048), sycl::range<3>(1, 1, 2048)),
      [=](sycl::nd_item<3> item_ct1) {
        k();
      });
}
```

which is rewritten to:

```
void k() {}

void foo() {
```

```
  size_t max_work_group_size =
      dpct::get_default_queue()
          .get_device()
          .get_info<sycl::info::device::max_work_group_size>();
  size_t work_group_size = 2048;
  if (work_group_size > max_work_group_size) {
    work_group_size = max_work_group_size;
  }
  size_t work_group_num = std::ceil((float)2048 / (float)work_group_size);
  dpct::get_default_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, work_group_num * work_group_size),
                        sycl::range<3>(1, 1, work_group_size)),
      [=](sycl::nd_item<3> item_ct1) { k(); });
}
```

DPCT1050

Message

The template argument of the *<type>* could not be deduced. You need to update this code.

Detailed Help

This warning is generated when the template argument could not be deduced by the Intel® DPC++ Compatibility Tool because the variable of this type was not used directly in the code. Intel® DPC++ Compatibility Tool inserts "dpct_placeholder", instead of type, in such cases.

Suggestions to Fix

Replace the "dpct_placeholder" with the real argument.

For example, this original CUDA* code:

```
__global__ void kernel(const cudaTextureObject_t texObj) {}

void foo() {
  float4 *d_data42;
  cudaArray_t a42;
  cudaMalloc(&d_data42, sizeof(float4) * 32 * 32);
  cudaChannelFormatDesc desc42 = cudaCreateChannelDesc(32, 32, 32, 32,
cudaChannelFormatKindFloat);
  cudaMallocArray(&a42, &desc42, 32, 32);
  cudaMemcpyToArray(a42, 0, 0, d_data42, 32 * 32 * sizeof(float4), cudaMemcpyDeviceToDevice);
  cudaTextureObject_t tex42;
  cudaResourceDesc res42;
  cudaTextureDesc texDesc42;
  res42.resType = cudaResourceTypeArray;
  res42.res.array.array = a42;
```

```
  cudaCreateTextureObject(&tex42, &res42, &texDesc42, NULL);
  kernel<<<1, 1>>>(tex42);
}
```

results in the following migrated SYCL* code:

```
/*
DPCT1050:1: The template argument of the image_accessor_ext could not be
deduced. You need to update this code.
*/
void kernel(const dpct::image_accessor_ext<
            dpct_placeholder /*Fix the type manually*/, 1>
                texObj) {}

void foo() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
  sycl::float4 *d_data42;
  dpct::image_matrix_p a42;
  d_data42 = (sycl::float4 *)sycl::malloc_device(sizeof(sycl::float4) * 32 * 32,
                                                 q_ct1);
  dpct::image_channel desc42 =
      dpct::image_channel(32, 32, 32, 32, dpct::image_channel_data_type::fp);
  a42 = new dpct::image_matrix(desc42, sycl::range<2>(32, 32));
  dpct::dpct_memcpy(a42->to_pitched_data(), sycl::id<3>(0, 0, 0),
                    dpct::pitched_data(d_data42, 32 * 32 * sizeof(sycl::float4),
                                       32 * 32 * sizeof(sycl::float4), 1),
                    sycl::id<3>(0, 0, 0),
                    sycl::range<3>(32 * 32 * sizeof(sycl::float4), 1, 1));
  dpct::image_wrapper_base_p tex42;
  dpct::image_data res42;
  dpct::sampling_info texDesc42;

  res42.set_data(a42);
  tex42 = dpct::create_image_wrapper(res42, texDesc42);
  /*
  DPCT1050:0: The template argument of the image_accessor_ext could not be
  deduced. You need to update this code.
  */
  q_ct1.submit([&](sycl::handler &cgh) {
    auto tex42_acc = static_cast<dpct::image_wrapper<
        dpct_placeholder /*Fix the type manually*/, 1> *>(tex42)
                         ->get_access(cgh);

    auto tex42_smpl = tex42->get_sampler();

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel(dpct::image_accessor_ext<
                 dpct_placeholder /*Fix the type manually*/, 1>(tex42_smpl,
                                                                tex42_acc));
        });
  });
}
```

which is rewritten to:

```
void kernel(const dpct::image_accessor_ext<sycl::float4, 2> texObj) {}

void foo() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
  sycl::float4 *d_data42;
  dpct::image_matrix_p a42;
  d_data42 = (sycl::float4 *)sycl::malloc_device(sizeof(sycl::float4) * 32 * 32,
                                                 q_ct1);
  dpct::image_channel desc42 =
      dpct::image_channel(32, 32, 32, 32, dpct::image_channel_data_type::fp);
  a42 = new dpct::image_matrix(desc42, sycl::range<2>(32, 32));
  dpct::dpct_memcpy(a42->to_pitched_data(), sycl::id<3>(0, 0, 0),
                    dpct::pitched_data(d_data42, 32 * 32 * sizeof(sycl::float4),
                                       32 * 32 * sizeof(sycl::float4), 1),
                    sycl::id<3>(0, 0, 0),
                    sycl::range<3>(32 * 32 * sizeof(sycl::float4), 1, 1));
  dpct::image_wrapper_base_p tex42;
  dpct::image_data res42;
  dpct::sampling_info texDesc42;

  res42.set_data(a42);
  tex42 = dpct::create_image_wrapper(res42, texDesc42);

  q_ct1.submit([&](sycl::handler &cgh) {
    auto tex42_acc =
        static_cast<dpct::image_wrapper<sycl::float4, 2> *>(tex42)->get_access(
            cgh);

    auto tex42_smpl = tex42->get_sampler();

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel(
              dpct::image_accessor_ext<sycl::float4, 2>(tex42_smpl, tex42_acc));
        });
  });
}
```

DPCT1051

Message

SYCL does not support a device property functionally compatible with <*property name*>. It was migrated to <*migrated token*>. You may need to adjust the value of <*migrated token*> for the specific device.

Detailed Help

Not all CUDA* device properties currently have functionally compatible equivalents in SYCL*. If such a property is detected and it is the integer type, it will be migrated to -1. If it is the Boolean type, it will be migrated to false or true depending on the context.

Suggestions to Fix

Review the logic and adjust it manually.

Visit the Migrating to SYCL forum for additional help.

For example, this original CUDA code:

```
void foo(cudaDeviceProp deviceProp) {
  size_t a = deviceProp.totalConstMem;
  ...
}
```

results in the following migrated SYCL code:

```
void foo(dpct::device_info deviceProp) {
  /*
  DPCT1051:0: SYCL does not support a device property functionally compatible
  with totalConstMem. It was migrated to get_global_mem_size. You may need to
  adjust the value of get_global_mem_size for the specific device.
  */
  size_t a = deviceProp.get_global_mem_size();
  ...
}
```

which is rewritten to:

```
void foo(dpct::device_info deviceProp) {
  size_t a = deviceProp.get_global_mem_size();
  ...
}
```

DPCT1052

Message

SYCL does not support the member access for a volatile qualified vector type. The volatile qualifier was removed. You may need to rewrite the code.

Detailed Help

The SYCL* specification does not provide a volatile qualified version of member access for vector type.

Suggestions to Fix

Review the code and adjust it.

For example, this original CUDA* code:

```
struct Complex : cuFloatComplex {
  float real() const volatile {
    return x;
  }
};
```

results in the following migrated SYCL code:

```
struct Complex : sycl::float2 {
  float real() const volatile {
    /*
    DPCT1052:0: SYCL does not support the member access for a volatile qualified
    vector type. The volatile qualifier was removed. You may need to rewrite the
    code.
    */
    return const_cast<struct Complex *>(this)->x();
  }
};
```

which is rewritten to:

```
struct Complex : sycl::float2 {
  float real() const volatile {
    return const_cast<struct Complex *>(this)->x();
  }
};
```

Visit the Migrating to SYCL forum for additional help.

DPCT1053

Message

Migration of device assembly code is not supported.

Detailed Help

The assembly code was not replaced because it is hardware-specific. Replace this hardware-specific feature with the one offered by your hardware vendor.

Suggestions to Fix

Consult with your hardware vendor to find a replacement.

DPCT1054

Message

The type of variable <*variable name*> is declared in device function with the name <*type*>. Adjust the code to make the <*type*> declaration visible at the accessor declaration point.

Detailed Help

This warning will be emitted when the type of a `__shared__` variable is declared in a device function.

In the migrated code:

- in the device function, the tool adds:

  # `uint8_t*` input parameter # name to the type, if unnamed, for example `type_ct1` # a type cast from `uint8_t*` to the original type, after the original type declaration.
- in the call side, when defining local accessor, the first template argument will be set to `uint8_t[sizeof(<original type, for example "type_ct1">)]`.

For example, this original CUDA* code:

```
// header file
template <typename T> __global__ void k() {
  __shared__ union {
    T t;
    ...
  } a;
  ...
}

// source file
void foo() { k<int><<<1, 1>>>(); }
```

results in the following migrated SYCL* code:

```
// header file
template <typename T> void k(uint8_t *a_ct1) {
  union type_ct1 {
    T t;
    ...
  };
  type_ct1 &a = *(type_ct1 *)a_ct1;
  ...
}

// source file
void foo() {
  dpct::get_default_queue().submit([&](sycl::handler &cgh) {
    /*
    DPCT1054:0: The type of variable a is declared in device function with the
    name type_ct1. Adjust the code to make the type_ct1 declaration visible at
    the accessor declaration point.
    */
    sycl::local_accessor<uint8_t[sizeof(type_ct1)], 0> a_ct1_acc_ct1(cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          k<int>(a_ct1_acc_ct1.get_pointer());
        });
  });
}
```

which is rewritten to:

```
// header file
template <typename T>
union type_ct1 {
  T t;
  ...
};
template <typename T> void k(uint8_t *a_ct1) {
  type_ct1<T> &a = *(type_ct1<T> *)a_ct1;
  ...
}

// source file
void foo() {
  dpct::get_default_queue().submit([&](sycl::handler &cgh) {
    sycl::local_accessor<uint8_t[sizeof(type_ct1<T>)], 0> a_ct1_acc_ct1(cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          k<int>(a_ct1_acc_ct1.get_pointer());
        });
  });
}
```

Suggestions to Fix

Move the type declaration so that it will be visible at the accessor declaration point or replace the `sizeof(<original type>` with size in bytes needed for the original type.

DPCT1055

Message

Vector types with size 1 are migrated to the corresponding fundamental types, which cannot be inherited. You need to rewrite the code.

Detailed Help

The warning message is emitted when vector type with size 1 is inherited by a class or struct in the original code. Since the vector type with size 1 is migrated to the corresponding fundamental type in DPC++ and the fundamental type cannot be inherited, you need to rewrite the code.

Suggestions to Fix

You can declare a new field with the corresponding fundamental type, for example int for int1, in the class/struct and override the required operators.

For example, this original CUDA* code:

```
class MyClass : int1 {
  ...
};
```

results in the following migrated SYCL* code:

```
/*
DPCT1055:0: Vector types with size 1 are migrated to the corresponding
fundamental types, which cannot be inherited. You need to rewrite the code.
*/
class MyClass : int {
  ...
};
```

which is rewritten to:

```
class MyClass {
public:
  int x;
  MyClass operator+(const MyClass& y) { ... }
  MyClass operator=(const MyClass& y) { ... }
  ...
};
```

DPCT1056

Message

The use of <*variable name*> in device code was not detected. If this variable is also used in device code, you need to rewrite the code.

Detailed Help

If `__constant__` variable is only used in host code, "__constant__" attribute will be removed.

Suggestions to Fix

For example, this original CUDA* code:

```
#include <stdio.h>
#include <cuda_runtime.h>

static __constant__ const float const_a = 10.f;
```

```
void foo_host() {
  printf("%f\n", const_a);
}
```

is migrated using the following migration command:

```
dpct --out-root out test.h --extra-arg=-xc
```

which results in the following migrated SYCL* code:

```
/*
DPCT1056:0: The use of const_a in device code was not detected. If this variable is
also used in device code, you need to rewrite the code.
*/
static const float const_a = 10.f;

void foo_host() {
  printf("%f\n", const_a);
}
```

If the variable “const_a” is only used in host code, just ignore the DPCT1056 warning. For example:

```
static const float const_a = 10.f;

void foo_host() {
  printf("%f\n", const_a);
}
```

If the variable “const_a” is used in both host and device code, rewrite the migrated code. For example:

```
static const float const_a_host_ct1 = 10.f;
static dpct::constant_memory<const float, 0> const_a(10.f);

void foo_host() {
  printf("%f\n", const_a_host_ct1);
}
```

DPCT1057

Message

Variable <*variable name*> was used in host code and device code. <*variable name*> type was updated to be used in SYCL device code and new <*host variable name*> was generated to be used in host code. You need to update the host code manually to use the new <*host variable name*>.

Detailed Help

If `__constant__variable` is used in both host code and device code (for example, the variable is included in two compilation units and they are compiled by different compilers), it will be migrated to a `dpct::constant_memory` object and a new host variable <*host variable name*>.

Suggestions to Fix

You need to update the host code manually to use the new *<host variable name>*.

For example, this original CUDA* code:

```
// h.h:
#include <cuda_runtime.h>
#include <stdio.h>

static __constant__ const float const_a = 10.f;

// a.cu:
#include "h.h"

__device__ void foo_device() {
  printf("%f\n", const_a);
}

// b.c:
#include "h.h"

void foo_host() {
  printf("%f\n", const_a);
}
```

is migrated using the following migration commands:

```
dpct --out-root out a.cu
dpct --out-root out b.c --extra-arg=-xc
```

which results in the following migrated SYCL* code:

```
// h.h:
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
#include <stdio.h>

/*
DPCT1057:0: Variable const_a was used in host code and device code. const_a type was
updated to be used in SYCL device code and new const_a_host_ct1 was generated to be
used in host code. You need to update the host code manually to use the new
const_a_host_ct1.
*/
static const float const_a_host_ct1 = 10.f;
static dpct::constant_memory<const float, 0> const_a(10.f);

// a.dp.cpp:
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
#include "h.h"

void foo_device(const sycl::stream &stream_ct1, const float const_a) {
  /*
  DPCT1015:0: Output needs adjustment.
  */
  stream_ct1 << "%f\n";
}

// b.c:
#include "h.h"
```

```
void foo_host() {
  printf("%f\n", const_a);
}
```

The migrated SYCL code is rewritten to address the DPCT1057 warning:

```
// h.h:
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
#include <stdio.h>

static const float const_a_host_ct1 = 10.f;
static dpct::constant_memory<const float, 0> const_a(10.f);

// a.dp.cpp:
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
#include "h.h"

void foo_device(const sycl::stream &stream_ct1, const float const_a) {
  stream_ct1 << const_a << "\n";
}

// b.c:
#include "h.h"

void foo_host() {
  printf("%f\n", const_a_host_ct1);
}
```

DPCT1058

Message

<API Name> is not migrated because it is not called in the code.

Detailed Help

This warning is emitted when atomic API calls are used in macro definition and this macro is not called in the code.

Suggestions to Fix

If you want this macro to be migrated by Intel® DPC++ Compatibility Tool, please use the macro that needs migration in your code and pass the code to the tool again. Otherwise you can update the code manually or ignore the warning.

For example, this original CUDA* code:

```
#define ATOMIC_ADD(x, v)  atomicAdd(&x, v);

void foo() {
  // There is no using of ATOMIC_ADD macro
}
```

results in the following migrated SYCL* code:

```
/*
DPCT1058:0: "atomicAdd" is not migrated because it is not called in the code.
*/
#define ATOMIC_ADD(x, v) atomicAdd(&x, v);

void foo() {
  // There is no using of ATOMIC_ADD macro
}
```

which is rewritten to:

```
void foo() {
  // There is no using of ATOMIC_ADD macro
}
```

DPCT1059

Message

SYCL only supports 4-channel image format. Adjust the code.

Detailed Help

SYCL* supports only 4-channel image format. The warning is emitted, when the tool generates code with unsupported image format, which corresponds to the original code. You can fix the resulting code by changing the image format. Note: suggested workaround may impact code performance.

For example, this original CUDA* code:

```
static texture<uint2, 2> tex;

__global__ void test_image() {
  uint2 tex_data = tex2D(tex, 0, 0);
}
void foo(uint2 *image_data) {
  ...
  test_image<<<1, 1>>>();
}
```

results in the following migrated SYCL code:

```
/*
DPCT1059:0: SYCL only supports 4-channel image format. Adjust the code.
*/
dpct::image_wrapper<sycl::uint2, 2> tex;

void test_image(dpct::image_accessor_ext<sycl::uint2, 2> tex) {
  sycl::uint2 tex_data = tex.read(0, 0);
}
void foo(sycl::uint2 *image_data) {
```

```
  ...
  dpct::get_default_queue().submit([&](sycl::handler &cgh) {
    auto tex_acc = tex.get_access(cgh);

    auto tex_smpl = tex.get_sampler();

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          test_image(
              dpct::image_accessor_ext<sycl::uint2, 2>(tex_smpl, tex_acc));
        });
  });
}
```

which is rewritten to:

```
dpct::image_wrapper<sycl::uint4, 2> tex;

void test_image(dpct::image_accessor_ext<sycl::uint4, 2> tex) {
  sycl::uint4 tex_data = tex.read(0, 0);
}
void foo(sycl::uint2 *image_data) {
  // (1) Please allocate sycl::uint4 type new_image_data and initialize the value with
image_data.
  // (2) Bind new_image_data to the tex object instead of the image_data.
  // (3) Free the memory of new_image_data after usage.
  ...
  dpct::get_default_queue().submit([&](sycl::handler &cgh) {
    auto tex_acc = tex.get_access(cgh);

    auto tex_smpl = tex.get_sampler();

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          test_image(
              dpct::image_accessor_ext<sycl::uint4, 2>(tex_smpl, tex_acc));
        });
  });
}
```

Suggestions to Fix

You may need to rewrite this code.

DPCT1060

Message

SYCL range can only be a 1D, 2D, or 3D vector. Adjust the code.

Detailed Help

This warning is emitted when the number of dimensions of memory in the original code exceeds 3. Since SYCL* range supports only 1, 2 or 3 dimensions, the resulting code is not SYCL-compliant.

To fix the resulting code you can use the low-dimensional arrays to simulate high-dimensional arrays.

The following fix example demonstrates how to use a 3D array to simulate a 4D array.

For example, this original CUDA* code:

```
__constant__ int const_array[dimX][dimY][dimZ][dimW];

__global__ void kernel(int x, int y, int z, int w) {
  int a = const_array[x][y][z][w];
}
```

results in the following migrated SYCL code:

```
/*
DPCT1060:0: SYCL range can only be a 1D, 2D, or 3D vector. Adjust the code.
*/
static dpct::constant_memory<int, 4> const_array(dimX, dimY, dimZ, dimW);

void kernel(int x, int y, int z, int w,
            sycl::accessor<int, 4, sycl::access_mode::read, sycl::access::target::device>
const_array) {
  int a = const_array[x][y][z][w];
}
```

which is rewritten to:

```
static dpct::constant_memory<int, 3> const_array(dimX, dimY, dimZ * dimW);

void kernel(int x, int y, int z, int w,
            sycl::accessor<int, 3, sycl::access_mode::read, sycl::access::target::device>
const_array) {
  int a = const_array[x][y][w * dimZ + z];
}
```

Suggestions to Fix

You may need to rewrite this code.

DPCT1061

Message

Call to <*macro name*> macro was removed because it only contains code, which is unnecessary in SYCL.

Detailed Help

When a macro definition contains only code, which will be removed during migration, Intel® DPC++ Compatibility Tool removes the usage of the macro as well.

Suggestions to Fix

Review the code and remove the macro definition if it is not needed.

For example, this original CUDA* code:

```
#define MACRO(x, y) cublasGetAtomicsMode(x, y)
void foo(cublasHandle_t h, cublasAtomicsMode_t *am) {
  MACRO(h, am);
  ...
}
```

results in the following migrated SYCL* code:

```
/*
DPCT1026:0: The call to cublasGetAtomicsMode was removed because this call is
redundant in SYCL.
*/
#define MACRO(x, y)
void foo(dpct::queue_ptr h, int *am) {
  /*
  DPCT1061:1: Call to MACRO macro was removed because it only contains code,
  which is unnecessary in SYCL.
  */
  ...
}
```

which is rewritten to:

```
void foo(dpct::queue_ptr h, int *am) {
  ...
}
```

DPCT1062

Message

SYCL Image doesn’t support normalized read mode.

Detailed Help

This warning is emitted when `cudaReadModeNormalizedFloat` is used as the third argument of texture in the original code. Since SYCL* Image doesn’t support normalized read mode, `cudaReadModeNormalizedFloat` will be ignored during migration.

It may cause errors in the resulting code, for example, redefinition of overloaded functions, if the overloaded functions are differentiated based on the texture type in the original code.

For example, this original CUDA* code:

```
__device__ void foo(const texture<char4, 2, cudaReadModeNormalizedFloat> tex) {
  float4 f = tex2D(tex, 0.5f, 0.5f);
}
__device__ void foo (const texture<char4, 2, cudaReadModeElementType> tex) {
  char4 c = tex2D(tex, 0.5f, 0.5f);
}
```

results in the following migrated SYCL code:

```
/*
DPCT1062:0: SYCL Image doesn't support normalized read mode.
*/
void foo(dpct::image_accessor_ext<sycl::char4, 2> tex) {
  sycl::float4 f = tex.read(0.5f, 0.5f);
}
void foo(dpct::image_accessor_ext<sycl::char4, 2> tex) {
  sycl::char4 c = tex.read(0.5f, 0.5f);
}
```

which is rewritten to:

```
void foo1(dpct::image_accessor_ext<sycl::char4, 2> tex) {
  sycl::char4 temp_c = tex.read(0.5f, 0.5f);
  sycl::float4 f{temp_c.x(), temp_c.y(), temp_c.z(), temp_c.w()};
  f = f / 255.f;
}
void foo2(dpct::image_accessor_ext<sycl::char4, 2> tex) {
  sycl::char4 c = tex.read(0.5f, 0.5f);
}
```

Suggestions to Fix

Review the code and update as needed.

DPCT1063

Message

Advice parameter is device-defined <*additional action*>. You may need to adjust it.

Detailed Help

The advice value is device-defined advice for the specified memory allocation. A value of 0 reverts the advice to the default behavior. Replace the value with the one required by your device.

Suggestions to Fix

Consult with your hardware vendor to find a replacement.

For example, this original CUDA* code:

```
void foo(int *devPtr, int device, size_t count) {
  cudaMemAdvise(devPtr, count, cudaMemAdviseSetReadMostly, device);
}
```

results in the following migrated SYCL* code:

```
void foo(int *devPtr, int device, size_t count) {
  /*
  DPCT1063:0: Advice parameter is device-defined and was set to 0. You may need
  to adjust it.
  */
  dpct::get_device(device).default_queue().mem_advise(devPtr, count, 0);
}
```

which is rewritten to:

```
void foo(int *devPtr, int device, size_t count) {
  dpct::get_device(device).default_queue().mem_advise(devPtr, count, 0);
}
```

DPCT1064

Message

Migrated <function name> call is used in a macro/template definition and may not be valid for all macro/template uses. Adjust the code.

Detailed Help

The warning is generated when the function call is used in a macro definition and needs to be migrated differently, depending on how the macro is called. The Intel® DPC++ Compatibility Tool generates code that is valid for one of the calls of the macro, but may not be valid for all calls of this macro in the code.

For example: The function `pow` can be migrated to `sycl::pow<double>` or `sycl::pown`, depending on the types of parameters passed through macro arguments.

For example, this original CUDA* code:

```
#define POW(B, E) pow(B, E)
__device__ void foo() {
  double a = POW(2.5, 3.1);
  double b = POW(2.5, 2);
}
```

results in the following migrated SYCL* code:

```
/*
DPCT1064:0: Migrated pow call is used in a macro/template definition and may not
be valid for all macro/template uses. Adjust the code.
*/
#define POW(B, E) dpct::pow(B, E)
void foo() {
```

```
  double a = POW(2.5, 3.1);
  double b = POW(2.5, 2);
}
```

which is rewritten to:

```
void foo() {
  double a = POW(2.5, 3.1);
  double b = 2.5 * 2.5;
}
```

Suggestions to Fix

Declare new macros for different use cases of the macro call in the resulting code.

DPCT1065

Message

Consider replacing `sycl::<...>::barrier()` with `sycl::<...>::barrier(sycl::access::fence_space::local_space)` for better performance if there is no access to global memory.

Detailed Help

The function `sycl::<...>::barrier()` ensures correct memory access ordering in the global and local address space. If the kernel function has no memory accesses in the global memory, it is safe to replace `sycl::<...>::barrier()` with `sycl::<...>::barrier(sycl::access::fence_space::local_space)` for better performance.

Suggestions to Fix

Replace `sycl::<...>::barrier()` with `sycl::<...>::barrier(sycl::access::fence_space::local_space)`.

For example, this original CUDA* code:

```
struct Data_t {
  float *host_data;
  float *device_data;
};

__global__ void k(Data_t *data) {
```

```
  auto tid = threadIdx.x + blockDim.x * blockIdx.x;
  only_read_data(data[tid].device_data);
  __syncthreads();
  only_read_data(data[tid].device_data);
}
```

results in the following migrated SYCL* code:

```
struct Data_t {
  float *host_data;
  float *device_data;
};

void k(Data_t *data, const sycl::nd_item<3> &item_ct1) {
  auto tid = item_ct1.get_local_id(2) +
             item_ct1.get_local_range(2) * item_ct1.get_group(2);
  only_read_data(data[tid].device_data);
  /*
  DPCT1065:0: Consider replacing sycl::nd_item::barrier() with
  sycl::nd_item::barrier(sycl::access::fence_space::local_space) for better
  performance if there is no access to global memory.
  */
  item_ct1.barrier();
  only_read_data(data[tid].device_data);
}
```

which is rewritten to:

```
struct Data_t {
  float *host_data;
  float *device_data;
};

void k(Data_t *data, const sycl::nd_item<3> &item_ct1) {
  auto tid = item_ct1.get_local_id(2) +
             item_ct1.get_local_range(2) * item_ct1.get_group(2);
  only_read_data(data[tid].device_data);
  // global_local_space can be replaced with local_space if the access
  // of the global memory after the barrier does not depend on (read-after-write or
  // write-after-read or write-after-write) the access of the same global memory
  // before the barrier among work-items in the current work-group.
  item_ct1.barrier(sycl::access::fence_space::local_space);
  only_read_data(data[tid].device_data);
}
```

DPCT1066

NOTE This diagnostic message is no longer generated by Intel® DPC++ Compatibility Tool 2023.0 or later.

Message

Input/Output distance are migrated to FWD/BWD_DISTANCE. You may need to update the code manually.

Detailed Help

When the tool cannot deduce the compute direction, the migration of Input/Output distance may need to be updated manually.

Suggestions to Fix

Update the migrated Input/Output distance according to the computing direction. If the direction is forward computing, Input/Output distance should map to FWD/BWD_DISTANCE. If the direction is backward computing, Input/Output distance should map to BWD/FWD_DISTANCE.

DPCT1067

Message

The <*parameter name*> parameter could not be migrated. You may need to update the code manually.

Detailed Help

Intel® oneAPI Collective Communications Library (oneCCL) currently does not support average reduction operation.

Suggestions to Fix

Rewrite the code manually.

DPCT1068

NOTE This diagnostic message is no longer generated by Intel® DPC++ Compatibility Tool 2023.2 or later.

Message

The value of <*argument type*> could not be deduced. You need to update <*placeholder*> manually.

Detailed Help

Intel® DPC++ Compatibility Tool could not deduce the value of the FFT type or dimensions. The value of dimension, `FWD_DISTANCE`, `BWD_DISTANCE`, `INPUT_STRIDES`, and `OUTPUT_STRIDES` needs to be updated manually.

Suggestions to Fix

Replace the placeholder with the correct value and add the necessary `set_value()` statements before `commit()`.

Refer to the descriptor<precision, domain>::set_value function for more information.

DPCT1069

Message

The argument *<argument name>* of the kernel function contains virtual pointer(s), which cannot be dereferenced. Try to migrate the code with `usm-level=restricted`.

Detailed Help

Intel® DPC++ Compatibility Tool cannot process inner virtual pointer(s) when using option `--usm-level=none`.

Suggestions to Fix

Re-migrate the code without specifing `--usm-level=none`.

For example, this original CUDA* code:

```
struct AAA {
  int *a;
};

__global__ void k(AAA obj) {
  obj.a[2] = 123;
}

void foo() {
  AAA obj;
  int *a;
  cudaMalloc(&a, sizeof(int) * 10);
```

```
  obj.a = a;
  k<<<1, 1>>>(obj);
}
```

results in the following migrated SYCL* code with option `--usm-level=none`:

```
#define DPCT_USM_LEVEL_NONE
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
struct AAA {
  int *a;
};

void k(AAA obj) {
  obj.a[2] = 123;
}

void foo() {
  AAA obj;
  int *a;
  a = (int *)dpct::dpct_malloc(sizeof(int) * 10);
  obj.a = a;
  /*
  DPCT1069:0: The argument 'obj' of the kernel function contains virtual
  pointer(s), which cannot be dereferenced. Try to migrate the code with
  "usm-level=restricted".
  */
  dpct::get_out_of_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
        k(obj);
      });
}
```

Re-migrated code without option `--usm-level=none`:

```
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
struct AAA {
  int *a;
};

void k(AAA obj) {
  obj.a[2] = 123;
}

void foo() {
  sycl::device dev_ct1;
  sycl::queue q_ct1(dev_ct1,
                    sycl::property_list{sycl::property::queue::in_order()});
  AAA obj;
  int *a;
  a = sycl::malloc_device<int>(10, q_ct1);
  obj.a = a;
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
```

```
        k(obj);
      });
}
```

DPCT1070

Message

<pointer variable name> is allocated by `dpct::dpct_malloc`. Use `dpct::get_host_ptr<type>(pointer variable name)` to access the pointer from the host code.

Detailed Help

The memory referenced by this pointer is allocated by `dpct::dpct_malloc` and cannot be directly accessed from the host code. You can access the memory from the host code by transforming the pointer using the `dpct::get_host_ptr` function.

Suggestions to Fix

1. Re-migrate the code without specifing `--usm-level=none`; or
2. Review the code and adjust it manually.

For example, this original CUDA* code:

```
void bar(float *a) {
  a[0] = 1;
}

void foo() {
  float* a;
  cudaMallocManaged(&a, 10 * sizeof(float));
  bar(a);
}
```

results in the following migrated SYCL* code:

```
#define DPCT_USM_LEVEL_NONE
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
void bar(float *a) {
  a[0] = 1;
}

void foo() {
  float* a;
  /*
  DPCT1070:0: 'a' is allocated by dpct::dpct_malloc. Use
  dpct::get_host_ptr<float>(a) to access the pointer from the host code.
  */
```

```
  a = (float *)dpct::dpct_malloc(10 * sizeof(float));
  bar(a);
}
```

which is rewritten to:

```
#define DPCT_USM_LEVEL_NONE
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>
void bar(float *a) {
  dpct::get_host_ptr<float>(a)[0] = 1;
}

void foo() {
  float* a;
  a = (float *)dpct::dpct_malloc(10 * sizeof(float));
  bar(a);
}
```

DPCT1071

NOTE This diagnostic message is no longer generated by Intel® DPC++ Compatibility Tool 2023.0 or later.

Message

The placement of the FFT computational function could not be deduced, so it is assumed out-of-place. You may need to adjust the code.

Detailed Help

If the last two arguments in the original API call are literally identical (ignoring C-Style type casts) and both the plan call and the execution call are in the same function, Intel® DPC++ Compatibility Tool generates in-place placement. Otherwise, it assumes out-of-place placement.

The difference of in-place and out-of-place may affect the value of `FWD_DISTANCE`, `BWD_DISTANCE`, `INPUT_STRIDES`, and `OUTPUT_STRIDES`.

Suggestions to Fix

If the placement is incorrect, you may need to manually add the necessary `set_value()` statements before the call to `commit()`.

Refer to the descriptor<precision, domain>::set_value function for more information.

DPCT1072

Message

SYCL currently does not support getting the available memory on the current device. You may need to adjust the code.

Detailed Help

SYCL* currently does not support getting the available memory on the current device.

Suggestions to Fix

Try to: (1) re-migrate the code without option `--no-dpcpp-extensions=device_info`; or (2) rewrite the code manually.

For example, this original CUDA* code:

```
void foo() {
  size_t result1, result2;
  cudaMemGetInfo(&result1, &result2);
}
```

results in the following migrated SYCL code:

```
void foo() {
  size_t result1, result2;
  /*
  DPCT1072:0: SYCL currently does not support getting the available memory on
  the current device. You may need to adjust the code.
  */
  result2 = dpct::get_current_device().get_device_info().get_global_mem_size();
}
```

which is rewritten to:

```
void foo() {
  size_t result1, result2;
  dpct::get_current_device().get_memory_info(result1, result2);
}
```

DPCT1073

Message

The field values of parameter <*parameter name*> could not be deduced, so the call was not migrated. You need to update this code manually.

Detailed Help

Intel® DPC++ Compatibility Tool could not deduce the field values of the call parameter, which is used in the migrated code.

Suggestions to Fix

Manually replace the non-migrated call with a DPC++ expression using the actual field values of the parameters.

For example, this original CUDA* code:

```
CUDA_ARRAY_DESCRIPTOR desc;

void bar(CUDA_ARRAY_DESCRIPTOR *desc_ptr) {
  CUarray arr;
  cuArrayCreate(&arr, desc_ptr);
}

void foo(CUarray_format f, size_t h, unsigned int n, size_t w) {
  desc.Height = h;
  desc.Width = w;
  desc.Format = f;
  desc.NumChannels = n;
  bar(&desc);
}
```

results in the following migrated SYCL* code:

```
size_t desc_x_ct1, desc_y_ct1;
unsigned desc_channel_num_ct1;
sycl::image_channel_type desc_channel_type_ct1;

/*
DPCT1082:1: Migration of CUDA_ARRAY_DESCRIPTOR * type is not supported.
*/
void bar(CUDA_ARRAY_DESCRIPTOR *desc_ptr) {
  dpct::image_matrix_p arr;
  /*
  DPCT1073:0: The field values of parameter 'desc_ptr' could not be deduced, so
  the call was not migrated. You need to update this code manually.
  */
  cuArrayCreate(&arr, desc_ptr);
}

void foo(sycl::image_channel_type f, size_t h, unsigned int n, size_t w) {
  desc_y_ct1 = h;
  desc_x_ct1 = w;
  desc_channel_type_ct1 = f;
  desc_channel_num_ct1 = n;
  bar(&desc);
}
```

which is manually adjusted to:

```
size_t desc_x_ct1, desc_y_ct1;
unsigned desc_channel_num_ct1;
sycl::image_channel_type desc_channel_type_ct1;

void bar() {
```

```
  dpct::image_matrix_p arr;
  arr = new dpct::image_matrix(desc_channel_type_ct1, desc_channel_num_ct1,
                               desc_x_ct1, desc_y_ct1);
}

void foo(sycl::image_channel_type f, size_t h, unsigned int n, size_t w) {
  desc_y_ct1 = h;
  desc_x_ct1 = w;
  desc_channel_type_ct1 = f;
  desc_channel_num_ct1 = n;
  bar();
}
```

DPCT1074

Message

The SYCL Image class does not support some of the flags used in the original code. Unsupported flags were ignored. Data read from SYCL Image could not be normalized as specified in the original code.

Detailed Help

Data read from **tex** in original code is normalized to a float in the range of (0, 1] by default, unless **CU_TRSF_READ_AS_INTEGER** is set. While data read from SYCL* image cannot be normalized to a float in the range of (0, 1] using the standard SYCL API. All flags used in the original code are ignored, except coordinate normalization mode.

Suggestions to Fix

Adjust the code manually.

For example, this original CUDA* code:

```
  // User does not set CU_TRSF_READ_AS_INTEGER here.
  cuTexRefSetFlags(tex, CU_TRSF_NORMALIZED_COORDINATES);
  cuTexRefSetFormat(tex, CU_AD_FORMAT_UNSIGNED_INT16, 1);
  ...
  result = tex2D<float>(tex, 0.5f, 0.5f);
```

results in the following migrated SYCL code:

```
  /*
  DPCT1074:0: The SYCL Image class does not support some of the flags used in
  the original code. Unsupported flags were ignored. Data read from SYCL Image
  could not be normalized as specified in the original code.
  */
  tex->set(sycl::coordinate_normalization_mode::normalized);
  tex->set_channel_type(sycl::image_channel_type::unsigned_int16);
  tex->set_channel_num(1);
  ...
  result = tex.read(0.5f, 0.5f); // data normalization does not match original code
```

which is manually adjusted to:

```
tex->set(sycl::coordinate_normalization_mode::normalized);
tex->set_channel_type(sycl::image_channel_type::unsigned_int16);
tex->set_channel_num(1);
...
result = tex.read(0.5f, 0.5f) / float(std::numeric_limits<std::uint16_t>::max());
```

DPCT1075

NOTE This diagnostic message is no longer generated by Intel® DPC++ Compatibility Tool 2023.0 or later.

Message

Migration of cuFFT calls may be incorrect and require review.

Detailed Help

The warning is generated in the following cases:

1. In the original code one cuFFT handle may be associated with different streams, but Intel® DPC++ Compatibility Tool could not detect which stream was used by cuFFT calls.
2. In the migrated code one or more commit calls need to be used, but the Intel® DPC++ Compatibility Tool could not deduce how to generate the correct set of commit calls.

For example, this original CUDA* code:

```
cufftPlan1d(plan, ...);
cufftExecR2C(plan, ...); // this call is using default stream
cufftSetStream(plan, s2);
cufftExecR2C(plan, ...); // this call is using s2 stream
```

results in the following migrated SYCL* code:

```
/*
DPCT1075:0: Migration of cuFFT calls may be incorrect and requires review.
*/
plan->commit(*s2); // wrong queue is used
...
dft::compute_forward (...);
...                                  // missing commit call
dft::compute_forward(...);
```

which is manually adjusted to:

```
plan->commit(dpct::get_default_queue()); // using default queue
...
dft::compute_forward (...);
...
plan->commit(*s2); // added new commit call with *s2 queue
dft::compute_forward(...);
```

Suggestions to Fix

Check that `oneapi::mkl::dft::descriptor::commit()` calls were generated correctly and are using the correct queue parameter. Fix the code if needed by adding missing commit calls and adjusting queue parameters.

Refer to the descriptor<precision, domain>::commit function for more information.

DPCT1076

Message

The device attribute was not recognized. You may need to adjust the code.

Detailed Help

This warning appears if the device attribute parameter in the original code is enum variable instead of enum constant and could not be deduced.

Suggestions to Fix

Review the code and adjust it.

For example, this original CUDA* code:

```
void foo(bool condition, int *val) {
  cudaDeviceAttr attr;
  if (condition)
    attr = cudaDevAttrHostNativeAtomicSupported;
  else
    attr = cudaDevAttrComputeCapabilityMajor;

  ...

  cudaDeviceGetAttribute(val, attr, 0);
}
```

results in the following migrated SYCL code:

```
void foo(bool condition, int *val) {
  int attr;
  if (condition)
    attr = 86;
  else
    attr = 75;

  ...

  /*
  DPCT1076:1: The device attribute was not recognized. You may need to adjust
```

```
  the code.
  */
  cudaDeviceGetAttribute(val, attr, 0);
}
```

which is rewritten to:

```
void foo(bool condition, int *val) {
  ...

  if (condition)
    *val = dpct::dev_mgr::instance().get_device(0).is_native_atomic_supported();
  else
    *val = dpct::dev_mgr::instance().get_device(0).get_major_version();
}
```

Visit the Migrating to SYCL forum for additional help.

DPCT1077

Message

<macro name> redefines a standard SYCL type, which may cause conflicts.

Detailed Help

Intel® DPC++ Compatibility Tool detected that the code redefines a standard SYCL* type, which may cause conflicts in the included SYCL header files.

Suggestions to Fix

Rename the macro to avoid conflicts with standard SYCL types.

For example, this original CUDA* code:

```
// test.h
inline void bar(cudaDeviceAttr attr) {
  ...
}

// test.cu
#define int2 int32_t
#include "test.h"

void foo() {
  int2 a;
  ...
}
```

results in the following migrated SYCL code:

```
// test.h
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>

inline void bar(int attr) {
  ...
}

// test.dp.cpp
/*
DPCT1077:0: 'int2' redefines a standard SYCL type, which may cause conflicts.
*/
#define int2 int32_t
#include "test.h"

void foo() {
  int2 a;
  ...
}
```

which is rewritten to:

```
// test.h
#include <sycl/sycl.hpp>
#include <dpct/dpct.hpp>

inline void bar(int attr) {
  ...
}

// test.dp.cpp
#include "test.h"

void foo() {
  int32_t a;
  ...
}
```

DPCT1078

Message

Consider replacing `memory_order::acq_rel` with `memory_order::seq_cst` for correctness if strong memory order restrictions are needed.

Detailed Help

`memory_order::acq_rel` is a light-weight fence that is sufficient for memory synchronization in most programs. If a program needs total sequentially consistent memory order to ensure correctness, replace `memory_order::acq_rel` with `memory_order::seq_cst`.

Suggestions to Fix

Replace `memory_order::acq_rel` with `memory_order::seq_cst` if stricter memory order is needed.

For example, this original CUDA* code:

```
__device__ void foo_dev() {
  ...
  __threadfence();
  ...
}
```

results in the following migrated SYCL code:

```
void foo_dev() {
  ...
  /*
  DPCT1078:0: Consider replacing memory_order::acq_rel with
  memory_order::seq_cst for correctness if strong memory order restrictions are
  needed.
  */
  sycl::atomic_fence(sycl::memory_order::acq_rel, sycl::memory_scope::device);
  ...
}
```

which is rewritten to:

```
void foo_dev() {
  ...
  // Assuming strong memory order restrictions are need here
  sycl::atomic_fence(sycl::memory_order::seq_cst, sycl::memory_scope::device);
  ...
}
```

DPCT1079

NOTE This diagnostic message is no longer generated by Intel® DPC++ Compatibility Tool 2023.2 or later.

Message

Replace “dpct_placeholder” with the file path of the dynamic library.

Detailed Help

The first argument of `dlopen()` or `LoadLibraryA()` should be the path to the dynamic library (for example `.so` on Linux* or `.dll` on Windows*).

Suggestions to Fix

Build the dynamic library from the migrated module file and replace “dpct_placeholder” with the file path.

DPCT1080

Message

Variadic functions cannot be called in a SYCL kernel or by functions called by the kernel. You may need to adjust the code.

Detailed Help

SYCL* device functions can’t call variadic functions.

Suggestions to Fix

Rewrite code manually by using non-variadic functions.

For example, this original CUDA* code:

```
__device__ int add(int arg_num, ...) {
  va_list args;
  va_start(args, arg_num);
  int sum = 0;
  for (int i = 0; i < arg_num; i++) {
    sum += va_arg(args, int);;
  }
  va_end(args);
  return sum;
};

__global__ void kernel(int *p) {
  int x = add(2, 123, 456);
  int y = add(3, 123, 456, 789);
}
```

results in the following migrated SYCL code:

```
/*
DPCT1080:0: Variadic functions cannot be called in a SYCL kernel or by functions
called by the kernel. You may need to adjust the code.
*/
int add(int arg_num, ...) {
  va_list args;
  va_start(args, arg_num);
  int sum = 0;
  for (int i = 0; i < arg_num; i++) {
    sum += va_arg(args, int);;
  }
  va_end(args);
  return sum;
```

```
};

void kernel(int *p) {
  int x = add(2, 123, 456);
  int y = add(3, 123, 456, 789);
}
```

which is rewritten to:

```
int add(int a0, int a1) {
  return a0 + a1;
};

int add(int a0, int a1, int a2) {
  return a0 + a1 + a2;
};

void kernel(int *p) {
  int x = add(123, 456);
  int y = add(123, 456, 789);
}
```

DPCT1081

Message

The generated code assumes that <*pointer variable*> points to the global memory address space. If it points to a local or private memory address space, replace `address_space::global` with `address_space::local` or `address_space::private`.

Detailed Help

Intel® DPC++ Compatibility Tool tries to deduce the address space of the memory that <*pointer variable*> points to. If the tool cannot deduce the address space (for example, the pointer variable is used by multiple functions), the tool will use `address_space::global` and the warning will be emitted.

Suggestions to Fix

Check the address space of the memory that <pointer variable> points to and replace `address_space::global` with the correct address space.

For example, this original CUDA* code:

```
__device__ void bar1(double *out1, double *d) {
  double i = 30;
  double &d2 = *d;
  sincos(i, out1, &d2);
}

__global__ void kernel(double *out1, double *out2) {
```

```
  double d;
  bar1(out1, &d);
  *out2 = d;
}
```

results in the following migrated SYCL code:

```
void bar1(double *out1, double *d) {
  double i = 30;
  double &d2 = *d;
  /*
  DPCT1081:0: The generated code assumes that "&d2" points to the global memory
  address space. If it points to a local or private memory address space,
  replace "address_space::global" with "address_space::local" or
  "address_space::private".
  */
  *out1 = sycl::sincos(
      i, sycl::address_space_cast<sycl::access::address_space::global_space,
                                  sycl::access::decorated::yes, double>(&d2));
}

void kernel(double *out1, double *out2) {
  double d;
  bar1(out1, &d);
  *out2 = d;
}
```

which is rewritten to:

```
void bar1(double *out1, double *d) {
  double i = 30;
  double &d2 = *d;
  *out1 = sycl::sincos(
      i, sycl::address_space_cast<sycl::access::address_space::private_space,
                                  sycl::access::decorated::yes, double>(&d2));
}

void kernel(double *out1, double *out2) {
  double d;
  bar1(out1, &d);
  *out2 = d;
}
```

DPCT1082

Message

Migration of <*type name*> type is not supported.

Detailed Help

Intel® DPC++ Compatibility Tool was unable to analyze and migrate the type <*type name*>.

Suggestions to Fix

This kind of code requires manual migration to DPC++.

Visit the Migrating to SYCL forum for additional help.

DPCT1083

Message

The size of *<placeholder>* in the migrated code may be different from the original code. Check that the allocated memory size in the migrated code is correct.

Detailed Help

Some types have a different size in the migrated code than in the original code, for example `sycl::float3` compared to `float3`. Check if the allocated size of memory is correct in the migrated code.

In the example below, `3*sizeof(float)` is used to represent the size of `float3` in the original code. In the migrated code the size of `sycl::float3` is different, so the allocated size needs adjustment.

For example, this original CUDA* code:

```
__global__ void kernel() {
  extern __shared__ float3 shared_memory[];
  ...
}

void foo() {
  size_t shared_size = 3 * sizeof(float);
  kernel<<<1, 1, shared_size>>>();
}
```

results in the following migrated SYCL* code:

```
void kernel(uint8_t *dpct_local) {
  auto shared_memory = (sycl::float3 *)dpct_local;
  ...
}

void foo() {
  /*
  DPCT1083:0: The size of local memory in the migrated code may be different
  from the original code. Check that the allocated memory size in the migrated
  code is correct.
  */
  size_t shared_size = 3 * sizeof(float);
  dpct::get_in_order_queue().submit([&](sycl::handler &cgh) {
    sycl::local_accessor<uint8_t, 1> dpct_local_acc_ct1(
        sycl::range<1>(shared_size), cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel(dpct_local_acc_ct1.get_pointer());
```

```
        });
  });
}
```

which is manually adjusted to:

```
void kernel(uint8_t *dpct_local) {
  auto shared_memory = (sycl::float3 *)dpct_local;
  ...
}

void foo() {
  size_t shared_size = 1 * sizeof(sycl::float3);
  dpct::get_in_order_queue().submit([&](sycl::handler &cgh) {
    sycl::local_accessor<uint8_t, 1> dpct_local_acc_ct1(
        sycl::range<1>(shared_size), cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel(dpct_local_acc_ct1.get_pointer());
        });
  });
}
```

Suggestions to Fix

Check the allocated size of memory and replace it with the correct size if necessary.

DPCT1084

Message

The function call <*function name*> has multiple migration results in different template instantiations that could not be unified. You may need to adjust the code.

Detailed Help

Intel® DPC++ Compatibility Tool was unable to migrate the code correctly. Modify the code manually.

The following example shows original code, migrated code, and the manual changes made to correct the migrated code.

For example, this original CUDA* code:

```
__constant__ int4 example_i[32];
__constant__ float4 example_f[32];

struct example_int {
  __device__ int4 foo(int idx) const { return example_i[idx]; }
};
```

```
struct example_float {
  __device__ float4 foo(int idx) const { return example_f[idx]; }
};

template <typename T> __global__ void example_kernel() {
  T example_v;
  int idx = blockIdx.x * blockDim.x + threadIdx.x;
  float j = example_v.foo(idx).x;
}

void foo() {
  example_kernel<example_int><<<1, 1>>>();
  example_kernel<example_float><<<1, 1>>>();
}
```

results in the following migrated SYCL* code:

```
static dpct::constant_memory<sycl::int4, 1> example_i(32);
static dpct::constant_memory<sycl::float4, 1> example_f(32);

struct example_int {
  sycl::int4 foo(int idx, sycl::int4 const *example_i) const {
    return example_i[idx];
  }
};

struct example_float {
  sycl::float4 foo(int idx, sycl::float4 const *example_f) const {
    return example_f[idx];
  }
};

template <typename T> void example_kernel(const sycl::nd_item<3> &item_ct1,
                                          sycl::float4 const *example_f) {
  T example_v;
  int idx = item_ct1.get_group(2) * item_ct1.get_local_range(2) +
            item_ct1.get_local_id(2);
  /*
  DPCT1084:0: The function call "example_int::foo" has multiple migration
  results in different template instantiations that could not be unified. You
  may need to adjust the code.
  */
  float j = example_v.foo(idx).x();
}

void foo() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.in_order_queue();
  q_ct1.submit([&](sycl::handler &cgh) {
    example_f.init();

    auto example_f_ptr_ct1 = example_f.get_ptr();

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          example_kernel<example_int>(item_ct1, example_f_ptr_ct1);
        });
```

```
  });
  q_ct1.submit([&](sycl::handler &cgh) {
    example_f.init();

    auto example_f_ptr_ct1 = example_f.get_ptr();

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          example_kernel<example_float>(item_ct1, example_f_ptr_ct1);
        });
  });
}
```

which is manually adjusted to:

```
static dpct::constant_memory<sycl::int4, 1> example_i(32);
static dpct::constant_memory<sycl::float4, 1> example_f(32);

struct example_int {
  typedef sycl::int4 data_type;
  sycl::int4 foo(int idx, sycl::int4 const *example_i) const {
    return example_i[idx];
  }
};

struct example_float {
  typedef sycl::float4 data_type;
  sycl::float4 foo(int idx, sycl::float4 const *example_f) const {
    return example_f[idx];
  }
};

template <typename T> void example_kernel(const sycl::nd_item<3> &item_ct1,
                                          typename T::data_type const *example) {
  T example_v;
  int idx = item_ct1.get_group(2) * item_ct1.get_local_range(2) +
            item_ct1.get_local_id(2);
  float j = example_v.foo(idx, example).x();
}

void foo() {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.in_order_queue();
  q_ct1.submit([&](sycl::handler &cgh) {
    example_i.init();

    auto example_i_ptr_ct1 = example_i.get_ptr();

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          example_kernel<example_int>(item_ct1, example_i_ptr_ct1);
        });
  });
  q_ct1.submit([&](sycl::handler &cgh) {
    example_f.init();

    auto example_f_ptr_ct1 = example_f.get_ptr();
```

```
    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          example_kernel<example_float>(item_ct1, example_f_ptr_ct1);
        });
  });
}
```

Suggestions to Fix

Code requires manual adjustment.

DPCT1085

Message

The function <*function name*> requires sub-group size to be <*size*>, while other sub-group functions in the same SYCL kernel require a different sub-group size. You may need to adjust the code.

Detailed Help

Each kernel can only be decorated with one sub-group size. This warning is emitted when a kernel requires different sub-group sizes. Check if the sub-group size can be unified into one value, and if it cannot be unified, redesign the code logic.

For example, this original CUDA* code:

```
__global__ void kernel(int* data1, int* data2) {
  typedef cub::WarpScan<int> WarpScan;
  typedef cub::WarpScan<int, 16> WarpScan16;

  typename WarpScan::TempStorage temp1;
  typename WarpScan16::TempStorage temp2;

  int input = data1[threadIdx.x];
  int output1 = 0;
  int output2 = 0;
  WarpScan(temp1).InclusiveSum(input, output1);
  data1[threadIdx.x] = output1;
  WarpScan16(temp2).InclusiveSum(input, output2);
  data2[threadIdx.x] = output1;
}

void foo(int* data1, int* data2) {
  kernel<<<1, 32>>>(data1, data2);
}
```

results in the following migrated SYCL* code:

```
void kernel(int* data1, int* data2, const sycl::nd_item<3> &item_ct1) {

  int input = data1[item_ct1.get_local_id(2)];
  int output1 = 0;
  int output2 = 0;
  output1 = sycl::inclusive_scan_over_group(item_ct1.get_sub_group(), input,
                                            sycl::plus<>());
  data1[item_ct1.get_local_id(2)] = output1;
  /*
  DPCT1085:0: The function inclusive_scan_over_group requires sub-group size to
  be 16, while other sub-group functions in the same SYCL kernel require a
  different sub-group size. You may need to adjust the code.
  */
  output2 = sycl::inclusive_scan_over_group(item_ct1.get_sub_group(), input,
                                            sycl::plus<>());
  data2[item_ct1.get_local_id(2)] = output1;
}

void foo(int* data1, int* data2) {
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 32), sycl::range<3>(1, 1, 32)),
      [=](sycl::nd_item<3> item_ct1) [[intel::reqd_sub_group_size(32)]] {
        kernel(data1, data2, item_ct1);
      });
}

void foo(int* data) {
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 32), sycl::range<3>(1, 1, 32)),
      [=](sycl::nd_item<3> item_ct1) [[intel::reqd_sub_group_size(32)]] {
        kernel(data, item_ct1);
      });
}
```

which is manually adjusted to:

```
void kernel(int* data1, int* data2, const sycl::nd_item<3> &item_ct1) {

  int input = data1[item_ct1.get_local_id(2)];
  int output1 = 0;
  int output2 = 0;
  output1 = sycl::inclusive_scan_over_group(item_ct1.get_sub_group(), input,
                                            sycl::plus<>());
  data1[item_ct1.get_local_id(2)] = output1;
  output2 = sycl::inclusive_scan_over_group(item_ct1.get_sub_group(), input,
                                            sycl::plus<>());
  data2[item_ct1.get_local_id(2)] = output1;
  item_ct1.barrier();
  if (item_ct1.get_local_id(2) % 32 >= 16) {
    int warp_id = item_ct1.get_local_id(2) / 32;
    data2[item_ct1.get_local_id(2)] -= data2[warp_id * 32 + 15];
  }
}

void foo(int* data1, int* data2) {
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 32), sycl::range<3>(1, 1, 32)),
      [=](sycl::nd_item<3> item_ct1) [[intel::reqd_sub_group_size(32)]] {
```

```
        kernel(data1, data2, item_ct1);
      });
}
```

Suggestions to Fix

Code requires manual fix. Rewrite the code manually.

DPCT1086

Message

`__activemask()` is migrated to `0xffffffff`. You may need to adjust the code.

Detailed Help

There is currently no functional equivalent of `__activemask()` in SYCL*. If there is flow control in your code that will make the thread inactive, you need to rewrite the thread logic.

For example, this original CUDA* code:

```
__device__ inline int SHFL_SYNC(unsigned mask, int val, unsigned offset,
                                unsigned w = warpSize) {
  return __shfl_down_sync(mask, val, offset, w);
}

__global__ void kernel(int *array) {
  unsigned int tid = threadIdx.x;
  if (tid >= 8)
    return;
  unsigned mask = __activemask();
  array[tid] = SHFL_SYNC(mask, array[tid], 4);
}
```

results in the following migrated SYCL code:

```
inline int SHFL_SYNC(unsigned mask, int val, unsigned offset,
                     const sycl::nd_item<3> &item_ct1, unsigned w = 0) {
  /*
  DPCT1023:0: The SYCL sub-group does not support mask options for
  dpct::shift_sub_group_left. You can specify
  "--use-experimental-features=masked-sub-group-operation" to use the
  experimental helper function to migrate __shfl_down_sync.
  */
  if (!w) w = item_ct1.get_sub_group().get_local_range().get(0);
  // This call will wait for all work-items to arrive which will never happen since only work-
items with tid < 8 will encounter this call.
  return dpct::shift_sub_group_left(item_ct1.get_sub_group(), val, offset, w);
}

void kernel(int *array, const sycl::nd_item<3> &item_ct1) {
```

```
  unsigned int tid = item_ct1.get_local_id(2);
  if (tid >= 8)
    return;
  /*
  DPCT1086:1: __activemask() is migrated to 0xffffffff. You may need to adjust
  the code.
  */
  unsigned mask = 0xffffffff;
  array[tid] = SHFL_SYNC(mask, array[tid], 4, item_ct1);
}
```

which is rewritten to:

```
// remove mask parameter, as it is not used
inline int SHFL_SYNC(int val, unsigned offset,
                     const sycl::nd_item<3> &item_ct1, unsigned w = 0) {
  if (!w) w = item_ct1.get_sub_group().get_local_range().get(0);
  unsigned int tid = item_ct1.get_local_id(2);
  // Use a temporary variable to save the result of sycl::shift_group_left() to make sure all
work-items can encounter this call.
  int v_tmp = sycl::shift_group_left(item_ct1.get_sub_group(), val, offset);
  return (tid < 8) ? v_tmp : val;
}

void kernel(int *array, const sycl::nd_item<3> &item_ct1) {
  unsigned int tid = item_ct1.get_local_id(2);
  // remove mask parameter, as it is not used
  array[tid] = SHFL_SYNC(array[tid], 4, item_ct1);
}
```

Suggestions to Fix

Check if `0xffffffff` can be used instead of `__activemask()`. If it cannot be used, redesign the thread logic.

DPCT1087 [UPDATE]

Message

SYCL currently does not support cross group synchronization. You can specify `--use-experimental-features=root-group` to use the root-group to migrate *<synchronization API call>*.

Detailed Help

By default, the SYCL root group extension is not used to migrate CUDA* grid level synchronization. To use `root-group` to migrate CUDA grid level synchronization, specify `--use-experimental-features=root-group` in the migration command.

Suggestions to Fix

Specify `--use-experimental-features= root-group` in the migration command to use the root-group to migrate CUDA grid level synchronization.

For example, this original CUDA* code:

```
__global__ void kernel() {
  namespace cg = cooperative_groups;
  cg::grid_group grid = cg::this_grid();
  grid.sync();
}

void foo() {
  kernel<<<1, 64>>>();
}
```

results in the following migrated SYCL code:

```
void kernel() {

  /*
  DPCT1119:1: Migration of cooperative_groups::__v1::grid_group::this_grid is not
  supported, please try to remigrate with option:
  --use-experimental-features=root-group.
  */
  /*
  DPCT1119:2: Migration of cooperative_groups::__v1::grid_group is not
  supported, please try to remigrate with option:
  --use-experimental-features=root-group.
  */
  cg::grid_group grid = cg::this_grid();
  /*
  DPCT1087:0: SYCL currently does not support cross group synchronization. You
  can specify "--use-experimental-features=root-group" to use the dpct
  helper function nd_range_barrier to migrate grid.sync().
  */
  grid.sync();
}

void foo() {
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 64), sycl::range<3>(1, 1, 64)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel();
      });
}
```

which is rewritten to:

```
void kernel(const sycl::nd_item<3> &item_ct1,
  sycl::ext::oneapi::experimental::root_group grid =
      item_ct1.ext_oneapi_get_root_group();
  sycl::group_barrier(grid);
}
```

```
void foo() {
  auto exp_props = sycl::ext::oneapi::experimental::properties{
      sycl::ext::oneapi::experimental::use_root_sync};
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 64), sycl::range<3>(1, 1, 64)),
      exp_props, [=](sycl::nd_item<3> item_ct1) {
        kernel(item_ct1);
          [=](sycl::nd_item<3> item_ct1) {
      });
}
```

DPCT1088

Message

The macro definition has multiple migration results in the dimension of free queries function that could not be unified. You may need to modify the code.

Detailed Help

Intel® DPC++ Compatibility Tool was unable to migrate the code correctly. You need to modify the code manually.

For example, this original CUDA* code:

```
namespace cg = cooperative_groups;

#define TB(b) cg::thread_block b = cg::this_thread_block();

__global__ void test1() {
  TB(b);
}

__global__ void test2() {
  TB(b);
}

void foo() {
  test1<<<dim3(2, 2, 2), dim3(2, 2, 2)>>>();
  test2<<<2, 2>>>();
}
```

migrated with options `--assume-nd-range-dim=1` and `--use-experimental-features=free-function-queries`, results in the following migrated SYCL code:

```
/*
DPCT1088:0: The macro definition has multiple migration results in the dimension
of free queries function that could not be unified. You may need to modify the
code.
*/
#define TB(b)                                                                  \
  auto b = sycl::ext::oneapi::this_work_item::get_work_group<                      \
      dpct_placeholder /* Fix the dimension manually */>();

void test1() {
  TB(b);
}
```

```
void test2() {
  TB(b);
}

void foo() {
  sycl::device dev_ct1;
  sycl::queue q_ct1(dev_ct1,
                    sycl::property_list{sycl::property::queue::in_order()});
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(2, 2, 2) * sycl::range<3>(2, 2, 2),
                        sycl::range<3>(2, 2, 2)),
      [=](sycl::nd_item<3> item_ct1) {
        test1();
      });
  q_ct1.parallel_for(sycl::nd_range<1>(sycl::range<1>(2) * sycl::range<1>(2),
                                       sycl::range<1>(2)),
                     [=](sycl::nd_item<1> item_ct1) {
                       test2();
                     });
}
```

which is manually adjusted to:

```
#define TB(b, d)                                                               \
  auto b = sycl::ext::oneapi::this_work_item::get_work_group<d>();

void test1() {
  TB(b, 3);
}

void test2() {
  TB(b, 1);
}

void foo() {
  sycl::device dev_ct1;
  sycl::queue q_ct1(dev_ct1,
                    sycl::property_list{sycl::property::queue::in_order()});
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(2, 2, 2) * sycl::range<3>(2, 2, 2),
                        sycl::range<3>(2, 2, 2)),
      [=](sycl::nd_item<3> item_ct1) {
        test1();
      });
  q_ct1.parallel_for(sycl::nd_range<1>(sycl::range<1>(2) * sycl::range<1>(2),
                                       sycl::range<1>(2)),
                     [=](sycl::nd_item<1> item_ct1) {
                       test2();
                     });
}
```

Suggestions to Fix

Rewrite the code manually.

DPCT1089

Message

The value of the sub-group size attribute argument *<argument name>* cannot be evaluated. Replace `dpct_placeholder` with an integral constant expression.

Detailed Help

The argument to the attribute must be an integral constant expression. This warning is emitted when Intel® DPC++ Compatibility Tool cannot evaluate the value of the sub-group size argument as an integral constant expression. Check if the sub-group size attribute argument can be replaced by an integral constant expression, and if not, redesign the code logic.

For example, this original CUDA* code:

```
  _global_ void kernel(int WarpSize) {   int Input, Output, Lane;   ...   Output = __shfl(Input,
Lane, WarpSize); }
  ...
  if(condition)
  {   kernel<<<GirdSize, BlockSize>>>(16); }
  else
  {   kernel<<<GirdSize, BlockSize>>>(32); }
```

results in the following migrated SYCL* code:

```
  void kernel(int WarpSize, sycl::nd_item<3> item_ct1) {
    int Input, Output, Lane;   ...   Output = Item_ct1.get_sub_group().shuffle(Input, Lane);
  }
  ...
  if(condition) {
    /* DPCT1089 */
    q_ct1.parallel_for(
      sycl::nd_range<3>(GridSize * BlockSize, BlockSize),
      [=](sycl::nd_item<3> item_ct1) [[intel::reqd_sub_group_size(dpct_placeholder)]]
{       kernel(16, item_ct1); });
  } else {
    /* DPCT1089 */
    q_ct1.parallel_for(
      sycl::nd_range<3>(GridSize * BlockSize, BlockSize),
      [=](sycl::nd_item<3> item_ct1) [[intel::reqd_sub_group_size(dpct_placeholder)]]
{       kernel(32, item_ct1); }
  }
```

which is manually adjusted to:

```
  void kernel(int WarpSize, sycl::nd_item<3> item_ct1) {   int Input, Output, Lane;   ...
Output = Item_ct1.get_sub_group().shuffle(Input, Lane); }
  ...
  if(condition) {
    /* DPCT1089 */
    q_ct1.parallel_for(
      sycl::nd_range<3>(GridSize * BlockSize, BlockSize),
      [=](sycl::nd_item<3> item_ct1) [[intel::reqd_sub_group_size(16)]]
    {       kernel(16, item_ct1); }
    );
```

```
  } else {
    /* DPCT1089 */
    q_ct1.parallel_for(
      sycl::nd_range<3>(GridSize * BlockSize, BlockSize),
      [=](sycl::nd_item<3> item_ct1) [[intel::reqd_sub_group_size(32)]]
    {       kernel(32, item_ct1); }
  }
```

Suggestions to Fix

Code requires manual fix. Replace “dpct_placeholder” with an integral constant expression.

DPCT1090

Message

SYCL does not support the device property that would be functionally compatible with <*property name*>. It was not migrated. You need to rewrite the code.

Detailed Help

Not all CUDA* device properties have functionally compatible equivalents in SYCL*. These device properties are not migrated.

Suggestions to Fix

Review and rewrite the code manually.

For example, this original CUDA* code:

```
void foo(int t) {
  cudaDeviceProp properties;
  cudaGetDeviceProperties(&properties, 0);
  int a = properties.regsPerBlock;
  if (a < t)
    code path 1
  else
    code path 2
}
```

results in the following migrated SYCL code:

```
void foo() {
  dpct::device_info properties;
  dpct::get_device_info(properties, dpct::dev_mgr::instance().get_device(0));
  /*
  DPCT1090:0: SYCL does not support the device property that would be
  functionally compatible with regsPerBlock. It was not migrated. You need to
  rewrite the code.
  */
  int a = properties.regsPerBlock;
  if (a < t)
    code path 1
  else
    code path 2
}
```

which is rewritten to:

```
void foo() {
  dpct::device_info properties;
  dpct::get_device_info(properties, dpct::dev_mgr::instance().get_device(0));

  // You need apply code path1 or path2 depending on the num of registers available on
  // the target hardware.
  code path 1/2
}
```

DPCT1091

Message

The function `dpct::segmented_reduce` only supports DPC++ native binary operation. Replace "dpct_placeholder" with a DPC++ native binary operation.

Detailed Help

`dpct::segmented_reduce` supports the following native binary operations:

- `sycl::plus`
- `sycl::bit_or`
- `sycl::bit_xor`
- `sycl::bit_and`
- `sycl::maximum`
- `sycl::minimum`
- `sycl::multiplies`

Suggestions to Fix

Review and rewrite the code manually.

For example, this original CUDA* code:

```
struct UserMin {
  template <typename T>
  __device__ __host__ __forceinline__ T operator()(const T &a,
                                                   const T &b) const {
    return (b < a) ? b : a;
  }
};

void foo(int num_segments, int *device_offsets, int *device_in, int *device_out,
         UserMin min_op, int initial_value) {
  size_t temp_storage_size;
  void *temp_storage = nullptr;

  cub::DeviceSegmentedReduce::Reduce(temp_storage, temp_storage_size, device_in,
                                     device_out, num_segments, device_offsets,
                                     device_offsets + 1, min_op, initial_value);

  cudaMalloc(&temp_storage, temp_storage_size);

  cub::DeviceSegmentedReduce::Reduce(temp_storage, temp_storage_size, device_in,
                                     device_out, num_segments, device_offsets,
                                     device_offsets + 1, min_op, initial_value);

  cudaDeviceSynchronize();
  cudaFree(temp_storage);
}
```

results in the following migrated SYCL code:

```
struct UserMin {
  template <typename T>
  __dpct_inline__ T operator()(const T &a, const T &b) const {
    return (b < a) ? b : a;
  }
};

void foo(int num_segments, int *device_offsets, int *device_in, int *device_out,
         UserMin min_op, int initial_value) {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.in_order_queue();

  /*
  DPCT1026:0: The call to cub::DeviceSegmentedReduce::Reduce was removed because
  this call is redundant in SYCL.
  */

  /*
  DPCT1092:1: Consider replacing work-group size 128 with different value for
  specific hardware for better performance.
  */
  /*
  DPCT1091:2: The function dpct::segmented_reduce only supports DPC++ native
  binary operation. Replace "dpct_placeholder" with a DPC++ native binary
  operation.
  */
  dpct::device::segmented_reduce<128>(
      q_ct1, device_in, device_out, num_segments, device_offsets,
      device_offsets + 1, dpct_placeholder, initial_value);
```

```
  dev_ct1.queues_wait_and_throw();
}
```

which is rewritten to:

```
void foo(int num_segments, int *device_offsets, int *device_in, int *device_out,
         UserMin min_op, int initial_value) {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.in_order_queue();

  int max_work_group_size = dev_ct1.get_max_work_group_size();
  if (max_work_group_size >= 256)
    dpct::device::segmented_reduce<256>(
        q_ct1, device_in, device_out, num_segments, device_offsets,
        device_offsets + 1, sycl::minimum(), initial_value);
  else
    dpct::device::segmented_reduce<128>(
        q_ct1, device_in, device_out, num_segments, device_offsets,
        device_offsets + 1, sycl::minimum(), initial_value);

  dev_ct1.queues_wait_and_throw();
}
```

DPCT1092

Message

Consider replacing work-group size 128 with different value for specific hardware for better performance.

Detailed Help

Replace work-group size 128 with a value of 256 or greater, if the hardware supports it.

Use function `dpct::device_info:: get_max_work_group_size()` to determine the maximum work-group size supported by the hardware.

Suggestions to Fix

Replace work-group size with a greater value for better performance, as supported by the hardware.

Refer to DPCT1091.

DPCT1093

Message

The *<Device ID>* device may be not the one intended for use. Adjust the selected device if needed.

Detailed Help

The `cudaSetDevice` function sets the device to run the code on. After migration to SYCL*, the logic for device selection may need to be updated.

For example, this original CUDA* code:

```
int main(int argc, char **argv) {
  cudaSetDevice(1); // Device 1 is the best choice in the original code.
  foo(argc, argv);
  ...
  return 0;
}
```

results in the following migrated SYCL code:

```
int main(int argc, char **argv) {
  /*
  DPCT1093:0: The "1" device may be not the one intended for use. Adjust the
  selected device if needed.
  */
  dpct::select_device(1); // Device 1 maybe not the best choice in SYCL.
  foo(argc, argv);
  ...
  return 0;
}
```

which is manually adjusted to:

```
int main(int argc, char **argv) {
  // User can check device list by command `sycl-ls` and filter the device by using
  // environment variable `ONEAPI_DEVICE_SELECTOR`.
  dpct::select_device(0);
  foo(argc, argv);
  ...
  return 0;
}
```

Suggestions to Fix

Review and adjust the device selection logic if needed.

DPCT1094

Message

Content contains misleading bidirectional Unicode characters.

Detailed Help

Content contains bidirectional Unicode characters that may cause the code or text to be interpreted differently than it appears.

Suggestions to Fix

You may need to remove bidirectional Unicode characters.

DPCT1095

Message

The identifier <*Identifier A*> is confusable with another identifier <*Identifier B*>.

Detailed Help

The identifier is confusable with another identifier in the same declaration scope and could lead to unexpected variable or function references.

A list of confusable identifiers can be found at https://www.unicode.org/Public/security/latest/confusables.txt.

Suggestions to Fix

You may need to check the original code.

DPCT1096

Message

The right-most dimension of the work-group used in the SYCL kernel that calls this function may be less than <*value of kernel sub-group size attribute*>. The function <*help function name*> may return an unexpected result on the CPU device. Modify the size of the work-group to ensure that the value of the right-most dimension is a multiple of <*value of kernel sub-group size attribute*>.

Detailed Help

The `dpct::select_from_sub_group`, `dpct::shift_sub_group_left`, `dpct::shift_sub_group_right`, and `dpct::permute_sub_group_by_xor` functions may return unexpected results when run on a CPU device with an OpenCL™ backend, if the right-most dimension value of the work-group used in the SYCL* kernel that calls these functions is less than the value of the kernel sub-group size attribute. The real sub-group size may not be the value specified by the kernel sub-group size attribute and could cause the helper function to return unexpected results on a CPU device.

Adjust the code by modifying the size of the work-group to ensure that the value of the right-most dimension is a multiple of the kernel sub-group size attribute.

For example, this original CUDA* code:

```
__global__ void kernel() {
  ...
  value = __shfl_down(x, delta);
  ...
}

void foo() {
  ...
  auto GridSize = dim3(2);
  auto BlockSize = dim3(8, 8, 1);
  kernel<<<GridSize, BlockSize>>>();
  ...
}
```

results in the following migrated SYCL code:

```
void kernel(const sycl::nd_item<3> &item_ct1) {
  ...
  /*
  DPCT1096:0: The right-most dimension of the work-group used in the SYCL kernel
  that calls this function may be less than "32". The function
  "dpct::shift_sub_group_left" may return an unexpected result on the CPU
  device. Modify the size of the work-group to ensure that the value of the
  right-most dimension is a multiple of "32".
  */
  value = dpct::shift_sub_group_left(item_ct1.get_sub_group(), x, delta); // May return
unexpected result on CPU
  ...
}

void foo() {
  ...
  auto GridSize = sycl::range<3>(1, 1, 2);
  auto BlockSize = sycl::range<3>(1, 8, 8); // Problem: value of the right-most dimension 8 is
less than the kernel sub group size attribute 32.
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(GridSize * BlockSize, BlockSize),
      [=](sycl::nd_item<3> item_ct1) [[intel::reqd_sub_group_size(32)]] {
        kernel(item_ct1);
      });
  ...
}
```

which is rewritten to:

```
void kernel(const sycl::nd_item<3> &item_ct1) {
  ...
  value = dpct::shift_sub_group_left(item_ct1.get_sub_group(), x, delta);
  ...
}

void foo() {
  ...
  auto GridSize = sycl::range<3>(1, 1, 2);
  auto BlockSize = sycl::range<3>(1, 2, 32); // Fix: modified work group size to make the right-
most dimension to be multiple of the kernel sub group size attribute value, which is 32.
```

```
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(GridSize * BlockSize, BlockSize),
      [=](sycl::nd_item<3> item_ct1) [[intel::reqd_sub_group_size(32)]] {
        kernel(item_ct1);
      });
  ...
}
```

Suggestions to Fix

If the program needs to execute on a CPU device, you may need to adjust the code.

DPCT1097

Message

The function <*backward function name*> may require the workspace used to save intermediate results from function <*forward function name*>. By default, a workspace from `engine_ext` is selected according to the source data pointer, but this may be incorrect and cause a workspace data race. You may need to rewrite this code.

Detailed Help

You can manually pass a `dnnl::memory` object generated from the forward function to the backward function.

For example, this original CUDA* code:

```
void test(cudnnHandle_t handle, cudnnTensorDescriptor_t dataTensor,
          cudnnTensorDescriptor_t outTensor,
          cudnnTensorDescriptor_t diffdataTensor,
          cudnnTensorDescriptor_t diffoutTensor, float *data, float *out,
          float *diffdata, float *diffout, float alpha, float beta,
          cudnnLRNDescriptor_t desc) {
  ...
  cudnnLRNCrossChannelForward(handle, desc, CUDNN_LRN_CROSS_CHANNEL_DIM1,
                              &alpha, dataTensor, data, &beta, outTensor, out);
  ...
  cudnnLRNCrossChannelBackward(handle, desc, CUDNN_LRN_CROSS_CHANNEL_DIM1,
                               &alpha, outTensor, out, diffoutTensor, diffout,
                               dataTensor, data, &beta, diffdataTensor,
                               diffdata);
  ...
}
```

results in the following migrated SYCL* code:

```
void test(dpct::dnnl::engine_ext handle, dpct::dnnl::memory_desc_ext dataTensor,
          dpct::dnnl::memory_desc_ext outTensor,
          dpct::dnnl::memory_desc_ext diffdataTensor,
          dpct::dnnl::memory_desc_ext diffoutTensor, float *data, float *out,
```

```
          float *diffdata, float *diffout, float alpha, float beta,
          dpct::dnnl::lrn_desc desc) {
  ...
  handle.async_lrn_forward(desc, alpha, dataTensor, data, beta, outTensor, out);
  ...
  /*
  DPCT1097:0: The function "async_lrn_backward" may require the workspace used
  to save intermediate results from function "async_lrn_forward". By default, a
  workspace from engine_ext is selected according to the source data pointer,
  but this may be incorrect and cause a workspace data race. You may need to
  rewrite this code.
  */
  handle.async_lrn_backward(desc, alpha, outTensor, out, diffoutTensor, diffout,
                            dataTensor, data, beta, diffdataTensor, diffdata);
  ...
}
```

which is manually adjusted to:

```
void test(dpct::dnnl::engine_ext handle, dpct::dnnl::memory_desc_ext dataTensor,
          dpct::dnnl::memory_desc_ext outTensor,
          dpct::dnnl::memory_desc_ext diffdataTensor,
          dpct::dnnl::memory_desc_ext diffoutTensor, float *data, float *out,
          float *diffdata, float *diffout, float alpha, float beta,
          dpct::dnnl::lrn_desc desc) {
  ...
  dnnl::memory workspace;
  handle.async_lrn_forward(desc, alpha, dataTensor, data, beta, outTensor, out,
                           &workspace);
  ...
  handle.async_lrn_backward(desc, alpha, outTensor, out, diffoutTensor, diffout,
                            dataTensor, data, beta, diffdataTensor, diffdata,
                            &workspace);
  ...
}
```

Suggestions to Fix

You may need to adjust the original code.

DPCT1098

Message

The <*expression text*> expression is used instead of the <*function name*> call. These two expressions do not provide the exact same functionality. Check the generated code for potential precision and/or performance issues.

Detailed Help

No exact functional equivalent of the function was found in SYCL* so an expression composed of other functions was used. Code precision and/or performance may not be as expected. Verify the code for correctness and performance.

Suggestions to Fix

Verify the code for correctness and performance.

For example, this original CUDA* code:

```
__device__ void device(float *f) {
  float a = __ldg(f);
}
```

results in the following migrated SYCL code:

```
void device(float *f) {
  /*
  DPCT1098:0: The '*' expression is used instead of the __ldg call. These two
  expressions do not provide the exact same functionality. Check the generated
  code for potential precision and/or performance issues.
  */
  float a = *f;
}
```

which is rewritten to:

```
void device(float *f) {
#if defined(__NVPTX__) && defined(__SYCL_DEVICE_ONLY__)
  float a = sycl::ext::oneapi::experimental::cuda::ldg(f);
#else
  float a = *f;
#endif
}
```

DPCT1099

Message

Verify if the default value of the direction and placement used in the function <*function name*> aligns with the related `compute` function call.

Detailed Help

By default, the parameter `direction_and_placement` is not passed for `dpct::fft::fft_engine::create()`, `dpct::fft::fft_engine::commit()`, or `dpct::fft::fft_engine::estimate_size()`. In this case, a forward direction (if the current FFT is complex-to-complex) and out-of-place (false) are used as the default values in this API. The configuration of

the direction and placement is reset when executing `dpct::fft::fft_engine::compute()` according to the argument passed with the `compute()` API. There might be performance issues caused by the reconfiguration.

To avoid the potential performance issues, specify the parameter `direction_and_placement` explicitly when calling `dpct::fft::fft_engine::create()` or `dpct::fft::fft_engine::commit()`, then the direction and placement information passed by the `compute()` API is ignored and no reconfiguration happens.

Suggestions to Fix

If you received a warning for `dpct::fft::fft_engine::create()` or `dpct::fft::fft_engine::commit()`:

- If you are using an internal workspace and do not care about performance, you can ignore this warning.
- If you do care about performance, or if you are using an external workspace, specify the `direction_and_placement` parameter correctly.

If you received this warning for `dpct::fft::fft_engine::estimate_size()`, you need to specify the `direction_and_placement` parameter correctly.

For example, this original CUDA* code:

```
void foo(cufftHandle plan, float2 *idata, float2 *odata) {
  size_t worksize;
  cufftMakePlan1d(plan, 7, CUFFT_C2C, 2, &worksize);
  cufftExecC2C(plan, idata, odata, CUFFT_INVERSE);
}
```

results in the following migrated SYCL code:

```
void foo(dpct::fft::fft_engine_ptr plan, sycl::float2 *idata,
         sycl::float2 *odata) {
  size_t worksize;
  /*
  DPCT1100:0: Currently the DFT external workspace feature in the Intel(R)
  oneAPI Math Kernel Library (oneMKL) is only supported on GPU devices. Use the
  internal workspace if your code should run on non-GPU devices.
  */
  /*
  DPCT1099:1: Verify if the default value of the direction and placement used in
  the function "commit" aligns with the related "compute" function call.
  */
  plan->commit(&dpct::get_in_order_queue(), 7,
               dpct::fft::fft_type::complex_float_to_complex_float, 2,
               &worksize);
  plan->compute<sycl::float2, sycl::float2>(idata, odata,
                                            dpct::fft::fft_direction::backward);
}
```

which is rewritten to:

```
void foo(dpct::fft::fft_engine_ptr plan, sycl::float2 *idata,
         sycl::float2 *odata) {
  // idata and odata are different memory
  size_t worksize;
  plan->commit(
      &dpct::get_in_order_queue(), 7,
      dpct::fft::fft_type::complex_float_to_complex_float, 2, &worksize,
      std::make_pair(
          dpct::fft::fft_direction::backward /*In the next compute() function,
             the direction is backward. So no need to change here.*/
          ,
          false /*is_inplace. In the next compute() function, idata and
                   odata are different memory, so set false here*/
          ));
  plan->compute<sycl::float2, sycl::float2>(idata, odata,
                                            dpct::fft::fft_direction::backward);
}
```

DPCT1100

Message

Currently the DFT external workspace feature in the Intel® oneAPI Math Kernel Library (oneMKL) is only supported on GPU devices. Use the internal workspace if your code should run on non-GPU devices.

Detailed Help

The FFT of Intel® oneAPI Math Kernel Library can use either an internal workspace or external workspace during computation. The internal workspace mode is supported on all devices, while the external workspace mode is currently only supported on GPU devices.

Suggestions to Fix

If you are running code on a non-GPU device, you can:

- Use `dpct::select_device(<device_id>)` to choose a GPU device.
- Call `dpct::fft::fft_engine::use_internal_workspace(true)` before invoking `dpct::fft::fft_engine::commit()`.

For example, this original CUDA* code:

```
void foo(cufftHandle plan, float2 *idata, float2 *odata) {
  size_t worksize;
  void *workspace;
  cufftSetAutoAllocation(plan, 0);
  cufftMakePlan1d(plan, 7, CUFFT_C2C, 2, &worksize);
  cudaMalloc(&workspace, worksize);
  cufftSetWorkArea(plan, workspace);
```

```
  cufftExecC2C(plan, idata, odata, CUFFT_FORWARD);
  ...
}
```

results in the following migrated SYCL code:

```
void foo(dpct::fft::fft_engine_ptr plan, sycl::float2 *idata,
         sycl::float2 *odata) {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.in_order_queue();
  size_t worksize;
  void *workspace;
  /*
  DPCT1100:0: Currently the DFT external workspace feature in the Intel(R)
  oneAPI Math Kernel Library (oneMKL) is only supported on GPU devices. Use the
  internal workspace if your code should run on non-GPU devices.
  */
  plan->use_internal_workspace(0);
  /*
  DPCT1100:1: Currently the DFT external workspace feature in the Intel(R)
  oneAPI Math Kernel Library (oneMKL) is only supported on GPU devices. Use the
  internal workspace if your code should run on non-GPU devices.
  */
  /*
  DPCT1099:2: Verify if the default value of the direction and placement used in
  the function "commit" is correct.
  */
  plan->commit(&q_ct1, 7, dpct::fft::fft_type::complex_float_to_complex_float,
               2, &worksize);
  workspace = (void *)sycl::malloc_device(worksize, q_ct1);
  /*
  DPCT1100:3: Currently the DFT external workspace feature in the Intel(R)
  oneAPI Math Kernel Library (oneMKL) is only supported on GPU devices. Use the
  internal workspace if your code should run on non-GPU devices.
  */
  plan->set_workspace(workspace);
  plan->compute<sycl::float2, sycl::float2>(idata, odata,
                                            dpct::fft::fft_direction::forward);
  ...
}
```

which is rewritten to:

```
void foo(dpct::fft::fft_engine_ptr plan, sycl::float2 *idata,
         sycl::float2 *odata) {
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.in_order_queue();
  size_t worksize;
  void *workspace;
  if (dev_ct1.is_gpu())
    plan->use_internal_workspace(0);
  plan->commit(&q_ct1, 7, dpct::fft::fft_type::complex_float_to_complex_float,
               2, &worksize);
  if (dev_ct1.is_gpu()) {
    workspace = (void *)sycl::malloc_device(worksize, q_ct1);
    plan->set_workspace(workspace);
  }
  plan->compute<sycl::float2, sycl::float2>(idata, odata,
```

```
                                          dpct::fft::fft_direction::forward);
  ...
}
```

DPCT1101

Message

<expression text> expression was replaced with a value. Modify the code to use the original expression, provided in comments, if it is correct.

Detailed Help

Identities, including constant variable names, types, and macros, used in the migrated SYCL* code are replaced with their value since they may not be available in the code scope. User can replace the value with the original expression listed in the comment manually after confirming that the identities are available in the code scope.

Suggestions to Fix

Use a macro or const variable to replace the value.

For example, this original CUDA* code:

```
__global__ void f() {
  const int x = 2;
  __shared__ int fold[x];
  ...
}

int main() {
  f<<<1, 1>>>();
  return 0;
}
```

results in the following migrated SYCL code:

```
void f(int *fold) {
  const int x = 2;
  ...
}

int main() {
  dpct::get_in_order_queue().submit([&](sycl::handler &cgh) {
    /*
    DPCT1101:0: 'x' expression was replaced with a value. Modify the code to
    use the original expression, provided in comments, if it is correct.
    */
    sycl::local_accessor<int, 1> fold_acc_ct1(sycl::range<1>(2 /*x*/), cgh);

    cgh.parallel_for(
```

```
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          f(fold_acc_ct1.get_pointer());
        });
  });
  return 0;
}
```

which is rewritten to:

```
const int x = 2;
void f(int *fold) {
  ...
}

int main() {
  dpct::get_in_order_queue().submit([&](sycl::handler &cgh) {
    sycl::local_accessor<int, 1> fold_acc_ct1(sycl::range<1>(x), cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          f(fold_acc_ct1.get_pointer());
        });
  });
  return 0;
}
```

DPCT1102

Message

Zero-length arrays are not permitted in SYCL device code.

Detailed Help

Zero-length arrays are not supported in the C++ standard or SYCL* specification. It may be supported by some host compiler implementations, but it is not supported by the Intel® oneAPI DPC++/C++ Compiler for device code.

Suggestions to Fix

Use an array with length greater than zero.

For example, this original CUDA* code:

```
#define DATA_NUM 16

struct Foo {
  ...
  float data[0];
```

```
};

__global__ void kernel(Foo *foo) {
  foo->data[DATA_NUM - 1] = 123.f;
}

void test() {
  Foo* foo;
  cudaMalloc(&foo, sizeof(Foo) + DATA_NUM * sizeof(float));
  kernel<<<1, 1>>>(foo);
  ...
}
```

results in the following migrated SYCL code:

```
#define DATA_NUM 16

struct Foo {
  ...
  /*
  DPCT1102:0: Zero-length arrays are not permitted in SYCL device code.
  */
  float data[0];
};

void kernel(Foo *foo) {
  foo->data[DATA_NUM - 1] = 123.f;
}

void test() {
  sycl::device dev_ct1;
  sycl::queue q_ct1(dev_ct1,
                    sycl::property_list{sycl::property::queue::in_order()});
  Foo* foo;
  foo =
      (Foo *)sycl::malloc_device(sizeof(Foo) + DATA_NUM * sizeof(float), q_ct1);
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel(foo);
      });
  ...
}
```

which is rewritten to:

```
#define DATA_NUM 16

struct Foo {
  ...
  std::byte data[DATA_NUM];
};

void kernel(Foo *foo) {
  foo->data[DATA_NUM - 1] = 123.f;
}

void test() {
  sycl::device dev_ct1;
```

```
    sycl::queue q_ct1(dev_ct1,
                      sycl::property_list{sycl::property::queue::in_order()});
    Foo* foo;
    foo = (Foo *)sycl::malloc_device(sizeof(Foo), q_ct1);
    q_ct1.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel(foo);
        });
    ...
  }
```

DPCT1103

Message

<expression text> should be a dynamic library. The dynamic library should supply wrapped kernel functions.

Detailed Help

The CUDA* application expects *<expression text>* to be the name of a CUDA module. The user should ensure that the CUDA code that generated the CUDA module is migrated to SYCL* code and compiled to generate a kernel library.

Suggestions to Fix

The user should find the module source files that are used to make the CUDA module and migrate them to SYCL using the `--extra-arg=-ptx` option of Intel® DPC++ Compatibility Tool to generate wrapped kernel functions:

```
dpct <module source file(s)> --extra-arg=--ptx
```

The migrated module source file(s) should be compiled to make a kernel library.

On Linux*:

```
icpx -fsycl <migrated module source file(s)> -fPIC -shared -o <kernel library filename>
```

On Windows*:

```
icpx -fsycl <migrated module source file(s)> -shared -o <kernel library filename>
```

The *<kernel library filename>* should be chosen so that *<expression text>* will have the same filename at runtime. For the following example, *<kernel library filename>* should be set to `library.so`.

For example, this original CUDA code:

```
void test() {
  CUmodule module;
  std::string ptxFile("module.ptx");
  cuModuleLoad(&module, ptxFile.c_str());
}
```

results in the following migrated SYCL code:

```
void test() {
  dpct::kernel_library module;
  std::string ptxFile("module.ptx");
  /*
  DPCT1103:0: 'ptxFile.c_str()' should be a dynamic library. The dynamic library
  should supply wrapped kernel functions.
  */
  module = dpct::load_kernel_library(ptxFile.c_str());
}
```

which is rewritten to:

```
void test() {
  dpct::kernel_library module;
  // "library.so" is compiled from the SYCL code which is migrated from CUDA code
  // used to build "module.ptx"
  std::string libraryFile("library.so");
  module = dpct::load_kernel_library(libraryFile);
}
```

DPCT1104

Message

<expression text> should point to a dynamic library loaded in memory. The dynamic library should supply wrapped kernel functions.

Detailed Help

The CUDA* application expects *<expression text>* to point to a CUDA module whose file image has been copied to memory. The user should ensure that the CUDA code that generated the CUDA module is migrated to SYCL* code, compiled to generate a kernel library, and then copied to memory by the migrated SYCL application.

Suggestions to Fix

The user should find the module source files that are used to make the CUDA module and migrate the module source files to SYCL using the `--extra-arg=--ptx` option of Intel® DPC++ Compatibility Tool to generate wrapped kernel functions:

```
dpct <module source file(s)> --extra-arg=--ptx
```

The migrated module source file(s) should be compiled to build a kernel library.

On Linux*:

```
icpx -fsycl <migrated module source file(s)> -fPIC -shared -o <kernel library filename>
```

On Windows*:

```
icpx -fsycl <migrated module source file(s)> -shared -o <kernel library filename>
```

The user should ensure that the application will load the kernel library image into memory at runtime. For the following example this can be ensured by setting <*kernel library filename*> to `library.so`.

For example, this original CUDA code:

```
void test() {
  CUmodule module;
  const void *image = loadFile("module.ptx");
  cuModuleLoadData(&module, image);
}
```

results in the following migrated SYCL code:

```
void test() {
  dpct::kernel_library module;
  const void *image = loadFile("module.ptx");
  /*
  DPCT1104:0: 'image' should point to a dynamic library loaded in memory. The
  dynamic library should supply wrapped kernel functions.
  */
  module = dpct::load_kernel_library_mem(image);
}
```

which is rewritten to:

```
void test() {
  dpct::kernel_library module;
  // "library.so" is compiled from the SYCL code which is migrated from CUDA code
  // used to build "module.ptx"
  const void *image = loadFile("library.so");
  module = dpct::load_kernel_library_mem(image);
}
```

DPCT1105

Message

The mcg59 random number generator is used. The subsequence argument <expression> is ignored. You need to verify the migration.

Detailed Help

The mcg59 random number generator does not support the subsequence parameter. If you want to use the subsequence parameter, you can use the philox4x32x10 random number generator instead.

Suggestions to Fix

For example, this original CUDA* code:

```
__global__ void kernel() {
  unsigned int u;
  curandStateXORWOW_t rng1;
```

```
  curand_init(1, 2, 3, &rng1);
  u = curand(&rng1);
}
```

results in the following migrated SYCL code:

```
void kernel() {
  /*
  DPCT1032:0: A different random number generator is used. You may need to adjust the code.
  */
  dpct::rng::device::rng_generator<oneapi::mkl::rng::device::mcg59<1>> rng1;
  /*
  DPCT1105:1: The mcg59 random number generator is used. The subsequence argument "2" is
ignored. You need to verify the    migration.
  */
  rng1 = dpct::rng::device::rng_generator<oneapi::mkl::rng::device::mcg59<1>>(1, 3);
  u = rng1.generate<oneapi::mkl::rng::device::uniform_bits<std::uint32_t>, 1>();
}
```

which is rewritten to:

```
void kernel() {
  dpct::rng::device::rng_generator<oneapi::mkl::rng::device::mcg59<1>> rng1;
  rng1 = dpct::rng::device::rng_generator<oneapi::mkl::rng::device::mcg59<1>>(1, 3);
  u = rng1.generate<oneapi::mkl::rng::device::uniform_bits<std::uint32_t>, 1>();
}
```

DPCT1106

Message

<function name> was migrated with the Intel extensions for device information which may not be supported by all compilers or runtimes. You may need to adjust the code.

Detailed Help

The CUDA* function *<function name>* (for example `cudaMemGetInfo` or `cuMemGetInfo`) was migrated with the Intel extensions for device information, which may not be supported by all compilers or runtimes. You may need to adjust the code.

For example, this original CUDA code:

```
  void foo() {
    size_t free_mem, total_mem;
    cudaMemGetInfo(&free_mem, &total_mem);
  }
```

results in the following migrated SYCL* code:

```
  void foo() {
    size_t free_mem, total_mem;
    /*
    DPCT1106:0: 'cudaMemGetInfo' was migrated with the Intel extensions for device
    information which may not be supported by all compilers or runtimes. You may
    need to adjust the code.
    */
    dpct::get_current_device().get_memory_info(free_mem, total_mem);
  }
```

which is rewritten to:

```
void foo() {
  size_t free_mem, total_mem;
  dpct::get_current_device().get_memory_info(free_mem, total_mem);
  // free_mem should be adjusted
  free_mem = /*implementation defined*/;
}
```

Suggestions to Fix

You may need to adjust the code.

DPCT1107

Message

Migration for this overload of <API Name> is not supported.

Detailed Help

Visit the Migrating to SYCL forum for additional help.

Suggestions to Fix

Visit the Migrating to SYCL forum for additional help.

DPCT1108

Message

<original API> was migrated with the experimental feature <feature name> which may not be supported by all compilers or runtimes. You may need to adjust the code.

Detailed Help

To support better migration, Intel® DPC++ Compatibility Tool provides the **-use-experimental-features** option to apply experimental features during migration. <feature name> may not be supported by all SYCL compilers or runtimes. If the target SYCL compiler and runtime does not support <feature name>, you need to adjust the code and not use <feature name>.

Suggestions to Fix

For example, this original CUDA* code:

```
__global__ void kernel() {
  int lane_id = threadIdx.x % 32;
  int foo = 0, result = 0;
  int mask = 0xf;
  if (lane_id == 0) {
    result = 10;
  }
  if (lane_id & mask) {
    foo = __shfl_sync(mask, result, 0);
  }
}
```

results in the following migrated SYCL code:

```
void kernel(const sycl::nd_item<3> &item_ct1) {
  int lane_id = item_ct1.get_local_id(2) % 32;
  int foo = 0, result = 0;
  int mask = 0xf;
  if (lane_id == 0) {
    result = 10;
  }
  if (lane_id & mask) {
    /*
    DPCT1108:0: '__shfl_sync' was migrated with the experimental feature masked
    sub_group function which may not be supported by all compilers or runtimes.
    You may need to adjust the code.
    */
    foo = dpct::experimental::select_from_sub_group(
        mask, item_ct1.get_sub_group(), result, 0);
  }
}
```

which is rewritten to:

```
void kernel(const sycl::nd_item<3> &item_ct1) {
  int lane_id = item_ct1.get_local_id(2) % 32;
  int foo = 0, result = 0;
  int mask = 0xf;
  if (lane_id == 0) {
    result = 10;
  }
  // If not use experimental feature masked sub-group function
  int foo_tmp = dpct::select_from_sub_group(item_ct1.get_sub_group(), result, 0);
  if (lane_id & mask) {
    foo = foo_tmp;
  }
}
```

DPCT1109

Message

<language feature> cannot be called in SYCL device code. You need to adjust the code.

Detailed Help

In SYCL device code, features like memory storage allocation, recursive function, and virtual function are not supported. Rewrite the code to comply with the SYCL specification.

Suggestions to Fix

For example, this original CUDA* code:

```
__global__ void kernel_func(size_t size) {
  int *data = new int[size];
  *data = 1;
}

void foo() {
  Kernel_func<<<1, 1>>>(size);
}
```

results in the following migrated SYCL code:

```
/*
DPCT1109:0: Memory storage allocation cannot be called in  SYCL device code. You need to adjust
the code.
*/
void kernel_func(size_t size) {
  int *data= new int[size];
  *data = 1;
}
void foo() {
  q.parallel_for(sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)), [=]
(sycl::nd_item<3> item_ct1) {kernel_func   (data, size); };)
}
```

which is rewritten to:

```
void kernel_func(int *data, size_t size) {
 *data = 1;
}

int *data= sycl::malloc_shared<int>(size, q); // Call the memory allocation function on the host
code.
q.parallel_for(sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)), [=]
(sycl::nd_item<3> item_ct1) {
  kernel_func(data, size);
};)
```

DPCT1110

Message

The total declared local variable size in device function <function name> exceeds 128 bytes and may cause high register pressure. Consult with your hardware vendor to find the total register size available and adjust the code, or use smaller sub-group size to avoid high register pressure.

Detailed Help

In specific hardware configurations, the number of registers available for each work-item is limited. For instance, in the Intel Xe-LP GPU architecture, each hardware thread has 4KB of registers. Consequently, if the sub-group size is 32, then each work-item can utilize 128 bytes of registers (4KB/32). If the declared local variable size in a device function exceeds 128 bytes, some variables may be stored in local or global memory, potentially leading to reduced performance when frequently accessed. To address this issue, you can either decrease the sub-group size to make more registers available for each work-item, or follow the recommendations in the Optimizing Register Spills section of the oneAPI GPU Optimization Guide. For other hardware, please consult with your hardware vendor to get configuration information.

Suggestions to Fix

For example, this original CUDA* code:

```
__global__ void Kernel(){
  int result[50];
  ...
}
int main{
  ...
  Kernel<<<1, 100>>>();
}
```

results in the following migrated SYCL* code:

```
/*
DPCT1110:0: The total declared local variable size in device function "Kernel" exceeds 128 bytes
and may cause high register pressure. Consult with your hardware vendor to find the total
register size available and adjust the code or use smaller    sub-group size to avoid high
register pressure.
*/
void Kernel(){
  int result[50];
  ...
}
int main{
  ...
  q.parallel_for(sycl::range(100), [=](sycl::nd_item<3> item) [[intel::reqd_sub_group_size(32)]]
{ Kernel(); });
}
```

which is rewritten to:

```
void Kernel(){
  int result[50];
  ...
}

int main{
  ...
  /*
   Reduce sub_group size to make more registers available for each work-item, which may help to
avoid high register pressure.
  */
  q.parallel_for(sycl::range(100), [=](sycl::nd_item<3> item) [[intel::reqd_sub_group_size(16)]]
{ Kernel(); });
}
```

DPCT1111

Message

Please verify the input arguments of <migrated API> base on the target function <kernel function name>.

Detailed Help

Intel® DPC++ Compatibility Tool cannot deduce some arguments from a function pointer or CUfunction variable. You need verify the argument value manually.

Suggestions to Fix

For example, this original CUDA* code:

```
extern __shared__ float a[];
__global__ void kernel() {
  __shared__ int b[10];
  __syncthreads();
}

void foo() {
  int numBlocks;
  cudaOccupancyMaxActiveBlocksPerMultiprocessor(&numBlocks, kernel, 128,
                                                sizeof(float) * 20);
  kernel<<<1, 128>>>();
}
```

results in the following migrated SYCL* code:

```
void kernel(const sycl::nd_item<3> &item_ct1, int *b) {

  item_ct1.barrier(sycl::access::fence_space::local_space);
}
```

```
void foo() {
  int numBlocks;
  /*
  DPCT1111:0: Please verify the input arguments of
  dpct::experimental::calculate_max_active_wg_per_xecore base on the target
  function "kernel".
  */
  dpct::experimental::calculate_max_active_wg_per_xecore(
      &numBlocks, 128,
      sizeof(float) * 20 +
          dpct_placeholder /* total shared local memory size */);
  dpct::get_default_queue().submit([&](sycl::handler &cgh) {
    sycl::local_accessor<int, 1> b_acc_ct1(sycl::range<1>(10), cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 128), sycl::range<3>(1, 1, 128)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel(item_ct1, b_acc_ct1.get_pointer());
        });
  });
}
```

which is rewritten to:

```
void kernel(const sycl::nd_item<3> &item_ct1, int *b) {
  item_ct1.barrier(sycl::access::fence_space::local_space);
}

void foo() {
  int numBlocks;
  dpct::experimental::calculate_max_active_wg_per_xecore(
      &numBlocks, 128, sizeof(float) * 20 + sizeof(int) * 10, 32, true, false);
  dpct::get_default_queue().submit([&](sycl::handler &cgh) {
    sycl::local_accessor<int, 1> b_acc_ct1(sycl::range<1>(10), cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 128), sycl::range<3>(1, 1, 128)),
        [=](sycl::nd_item<3> item_ct1) [[intel::reqd_sub_group_size(32)]] {
          kernel(item_ct1, b_acc_ct1.get_pointer());
        });
  });
}
```

DPCT1112

Message

If the filter mode is set to **linear**, the behavior of image “read” may be different from **tex1Dfetch** in the original code. You may need to adjust the code.

Detailed Help

The CUDA* **tex1Dfetch** function always applies the nearest filtering mode. In the migrated code, the image read function applies the filtering mode that is set for the texture object. There may be a different result on the image read, so you may need to align the filter mode on image read.

Suggestions to Fix

For example, this original CUDA code:

```
cudaTextureDesc desc;
desc.filterMode = cudaFilterModeLinear;
...
cudaTextureObject_t obj;
cudaCreateTextureObject(&obj, ..., &desc, ...);
...
tex1Dfetch(..., obj, ...);
```

results in the following migrated SYCL* code:

```
dpct::sampling_info desc;
desc.set(sycl::filtering_mode::linear);
...
dpct::image_wrapper_base_p obj;
obj = dpct::create_image_wrapper(..., desc);
...
/*
DPCT1112:0: If the filter mode is set to 'linear', the behavior of image
"read" may be different from "tex1Dfetch" in the original code. You may need
to adjust the code.
*/
obj.read(...);
```

which is rewritten to:

```
dpct::sampling_info desc;
desc.set(sycl::filtering_mode::nearest);  // Update the filtering mode from linear to nearest
...
dpct::image_wrapper_base_p obj;
obj = dpct::create_image_wrapper(..., desc);
...
```

DPCT1113

Message

Consider replacing `sycl::nd_item::barrier(sycl::access::fence_space::local_space)` with `sycl::nd_item::barrier()` if function <*function name*> is called in a multidimensional kernel.

Detailed Help

If there is no overlap global device memory access among each work-item in a work-group, the `__syncthreads()` API calls in kernel with 1D index space of a kernel execution that can be migrated to `sycl::nd_item::barrier(sycl::access::fence_space::local_space)`. If you call the kernel with 2D/3D index space, the global memory access from each work-item in a work-group may overlap and may result in data dependency between work-items across the barrier. If this is the case, you may need to update the `sycl::nd_item::barrier()` call with `sycl::access::fence_space::global_and_local`.

Suggestions to Fix

For example, this original CUDA* code:

```
__global__ void kernel(float *mem) {
  unsigned int tid = threadIdx.x + blockDim.x * blockIdx.x;
  ...
  mem[tid] = ...;
  __syncthreads();
  ...
  ... = mem[tid];
}

void foo(float *mem) {
  kernel<<<16, 16>>>(mem); //1D index space of a SYCL kernel execution
}
```

results in the following migrated SYCL* code:

```
void kernel(float *mem, const sycl::nd_item<3> &item_ct1) {
  unsigned int tid = item_ct1.get_local_id(2) +
                     item_ct1.get_local_range(2) * item_ct1.get_group(2);
  ...
  mem[tid] = ...; // global memory access without overlap among each work-item in a work-group
  /*
  DPCT1113:0: Consider replacing
  sycl::nd_item::barrier(sycl::access::fence_space::local_space) with
  sycl::nd_item::barrier() if function "kernel" is called in a multidimensional
  kernel.
  */
  item_ct1.barrier(sycl::access::fence_space::local_space);
  ...
  ... = mem[tid]; // global memory access without overlap among each work-item in a work-group
}

void foo(float *mem) {
  dpct::get_default_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 16) * sycl::range<3>(1, 1, 16),
                        sycl::range<3>(1, 1, 16)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel(mem, item_ct1);
      });
}
```

After migration, update the migrated SYCL kernel code to a 2D kernel:

```
void kernel(float *mem, const sycl::nd_item<3> &item_ct1) {
  unsigned int tidx = item_ct1.get_local_id(2) +
                      item_ct1.get_local_range(2) * item_ct1.get_group(2);
  unsigned int tidy = item_ct1.get_local_id(1) +
                      item_ct1.get_local_range(1) * item_ct1.get_group(1);

  ...
```

```
  mem[tidx] = ...; // global memory access with overlap among each work-item in a work-group
  mem[tidy] = ...; // global memory access with overlap among each work-item in a work-group
  /*
  DPCT1113:0: Consider replacing
  sycl::nd_item::barrier(sycl::access::fence_space::local_space) with
  sycl::nd_item::barrier() if function "kernel" is called in a multidimensional
  kernel.
  */
  item_ct1.barrier(sycl::access::fence_space::local_space);
  ...
  ... = mem[tidx]; // global memory access with overlap among each work-item in a work-group
  ... = mem[tidy]; // global memory access with overlap among each work-item in a work-group
}

void foo(float *mem) {
  dpct::get_default_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 4, 4) * sycl::range<3>(1, 4, 4),
                        sycl::range<3>(1, 4, 4)), /*2D  index space of a SYCL kernel execution */

      [=](sycl::nd_item<3> item_ct1) {
        kernel(mem, item_ct1);
      });
}
```

which is rewritten to:

```
void kernel(float *mem, const sycl::nd_item<3> &item_ct1) {
  unsigned int tidx = item_ct1.get_local_id(2) +
                      item_ct1.get_local_range(2) * item_ct1.get_group(2);
  unsigned int tidy = item_ct1.get_local_id(1) +
                      item_ct1.get_local_range(1) * item_ct1.get_group(1);

  ...
  mem[tidx] = ...; // global memory access with overlap among each work-item in a work-group
  mem[tidy] = ...; // global memory access with overlap among each work-item in a work-group
  item_ct1.barrier(sycl::access::fence_space::global_and_local);
  ...
  ... = mem[tidx]; // global memory access with overlap among each work-item in a work-group
  ... = mem[tidy]; // global memory access with overlap among each work-item in a work-group
}

void foo(float *mem) {
  dpct::get_default_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 4, 4) * sycl::range<3>(1, 4, 4),
                        sycl::range<3>(1, 4, 4)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel(mem, item_ct1);
      });
}
```

DPCT1114

Message

`cudaMemcpy` is migrated to asynchronization `memcpy`, assuming in the original code the source host memory is pageable memory. If the memory is not pageable, call `wait()` on event return by `memcpy` API to ensure synchronization behavior.

Detailed Help

The `cudaMemcpy` function typically operates synchronously. However, when copying from host to device using pageable host memory, its behavior becomes asynchronous. If the `--optimize-migration` option is used during migration, the migration tool assumes host memory is pageable and migrates `cudaMemcpy` into an asynchronous `memcpy` from host to device, which can improve performance by permitting concurrent memory transfer with other tasks. Nonetheless, if the source memory is pinned host memory, the user needs to call `wait()` on the event returned by the `memcpy` API to ensure synchronization behavior.

Suggestions to Fix

For example, this original CUDA* code:

```
Int N = 100;
float *src, *dst;
cudaMalloc(&dst, sizeof(float) * N);
cudaMallocHost(&src, sizeof(float) * N);
for(int i = 0; i < N; i++){
  Src[i] = i;
}
cudaMemcpy(dst, src, sizeof(float) * N, cudaMemcpyHostToDevice);
```

results in the following migrated SYCL* code:

```
sycl::device dev_ct1;
sycl::queue q_ct1(dev_ct1, sycl::property_list{sycl::property::queue::in_order()});
float *src, *dst;
dst = sycl::malloc_device<float>(N, q_ct1);
src = sycl::malloc_host<float>(N, q_ct1);
for(int i = 0; i < N; i++){
  src[i] = i;
}
/*
DPCT1114:1: cudaMemcpy is migrated to asynchronization memcpy, assuming in the original code the
source host memory is pageable memory. If  the memory is not pageable, call wait() on event
return by memcpy API to ensure synchronization behavior.
*/
q_ct1.memcpy(dst, src, sizeof(float) * N);
```

which is rewritten to:

```
sycl::device dev_ct1;
sycl::queue q_ct1(dev_ct1, sycl::property_list{sycl::property::queue::in_order()});
float *src, *dst;
dst = sycl::malloc_device<float>(N, q_ct1);
src = sycl::malloc_host<float>(N, q_ct1);
for(int i = 0; i < N; i++){
  src[i] = i;
}
q_ct1.memcpy(dst, src, sizeof(float) * N).wait(); // src is allocated by cudaMallocHost with
page-locked memory on host, so call wait().
```

DPCT1115

Message

The `sycl::ext::oneapi::group_local_memory_for_overwrite` is used to allocate group-local memory at the none kernel functor scope of a work-group data parallel kernel. You may need to adjust the code.

Detailed Help

The `sycl::ext::oneapi::group_local_memory_for_overwrite` can be used to allocate group-local memory at the kernel functor scope of a work-group data parallel kernel. The restriction that group-local variables must be defined at kernel functor scope may be lifted in a future version of this extension.

Refer to sycl_ext_oneapi_local_memory.asciidoc for more details.

Suggestions to Fix

For example, this original CUDA* code:

```
template <int S> __device__ void devfun() {
  __shared__ int slm1[32 * S];
  ...
}

template <int S> __global__ void kernel() {
  __shared__ int slm2[S];
  devfun<S>();
}

void hostfun() { kernel<256><<<1, 1>>>(); }
```

results in the following migrated SYCL* code:

```
template <int S> inline void devfun(int *p, const sycl::nd_item<3> &item_ct1) {
  /*
  DPCT1115:0: The sycl::ext::oneapi::group_local_memory_for_overwrite is used to allocate
  group-local memory at the none kernel functor scope of a work-group data
  parallel kernel. You may need to adjust the code.
  */
  auto &slm1 =
      *sycl::ext::oneapi::group_local_memory_for_overwrite<int[32 * S]>(item_ct1.get_group());
  ...
}

template <int S> __dpct_inline__ void kernel(const sycl::nd_item<3> &item_ct1) {
  auto &slm2 =
      *sycl::ext::oneapi::group_local_memory_for_overwrite<int[S]>(item_ct1.get_group());
  devfun<S>(item_ct1);
}
```

```
void hostfun() { dpct::get_default_queue().parallel_for(
  sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
  [=](sycl::nd_item<3> item_ct1) {
    kernel<256>(item_ct1);
  });
}
```

which is rewritten to:

```
template <int S> inline void devfun(int *slm1) {
  ...
}

template <int S> __dpct_inline__ void kernel(int *slm1, int *slm2) {

  devfun<S>(slm1);
}

void hostfun() { dpct::get_default_queue().submit(
  [&](sycl::handler &cgh) {
    sycl::local_accessor<int, 1> slm1_acc_ct1(sycl::range<1>(32 * 256), cgh);
    sycl::local_accessor<int, 1> slm2_acc_ct1(sycl::range<1>(256), cgh);

    cgh.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel<256>(slm1_acc_ct1.get_pointer(), slm2_acc_ct1.get_pointer());
      });
  });
```

DPCT1116

Message

The <*original API*> was migrated to <*migrated API*> for performance, and <*expression*> is computed by `(UINT_MAX + 1) / ('<argument>' + 1)`. This migration requires the initial value of <*value*> to be scaled by multiplying <*expression*>, and any usage of value of <*value*> outside atomic function to be scaled by dividing <*expression*>.

Detailed Help

By default, the migration tool uses the Compare-and-Swap atomic operation to migrate the `atomicInc` and `atomicDec` functions. If the atomic API is in a critical path, performance is not good.

To improve the performance, the migration tool checks the special value of the second argument of `atomicInc`/`atomicDec`. If it is `2n - 1 (1 < n <= 32)`, then the tool uses the atomic `add` API to migrate `atomicInc`/`atomicDec`.

Using `atomicInc(addr, val)` as an example, the essence of optimization involves:

- Migrate the `atomicInc(addr, val)` function to `dpct::atomic_fetch_add(addr, step)`, where `step` is calculated by `(UINT_MAX + 1) / (val + 1)`. This strategy exploits the overflow behavior of unsigned integers to mimic the wrapping behavior of `atomicInc/Dec()`.
- Update the initial value of `*addr` to be scaled by multiplying with `step`, ensuring the wrapping behavior is consistent.
- Update the references to the value `*addr` outside atomic functions, dividing it by `step` to retrieve the intended value.

Suggestions to Fix

For example, this original CUDA* code:

```
__device__ unsigned int a1 = 5;
__global__ void kernel(unsigned int *result){
  ...
  unsigned int old_val = atomicInc(&a1, 0x7fffffff);
  ...
  *result = a1;
}
```

results in the following migrated SYCL* code:

```
dpct::global_memory<unsigned int, 0> a1(5 * 2);
void kernel(unsigned int*result, unsigned int &a1){
  ...
  /*
  DPCT1116:0: The atomicInc was migrated to dpct::atomic_fetch_add(&a1, 2) / 2 for performance,
and 2 is computed by (UINT_MAX + 1) / ('0x7fffffff' + 1). This migration requires the initial
value of 'a1' to be scaled by multiplying 2, and any usage of the value of 'a1' outside atomic
function to be scaled by dividing 2.
  */
  unsigned int old_val = dpct::atomic_fetch_add<sycl::access::address_space::generic_space>(&a1,
2) / 2;
  ...
  *result = a1 / 2; // Dividing a1 by step 2 to get the intended value.
}
```

which is rewritten to:

```
dpct::global_memory<unsigned int, 0> a1(5 * 2);
void kernel(unsigned int*result, unsigned int &a1){
  ...
  unsigned int old_val = dpct::atomic_fetch_add<sycl::access::address_space::generic_space>(&a1,
2) / 2;
  ...
  *result = a1 / 2; // Dividing a1 by step 2 to get the intended value.
}
```

DPCT1117

Message

There is `atomicInc/Dec` operation on *<argument 2>* and value of *<argument 2>* was scaled by migration for performance, refer to DPCT1116. Using the value of *<argument 2>* through *<argument 1>* should also be scaled by dividing or multiplying *<expression>*. You may need to adjust the code.

Detailed Help

An optimized method has been applied to migrate `atomicInc`/`atomicDec`. To align with this optimized method, some scaling adjustments are required to ensure that values remain consistent and accurate after migration. Refer to DPCT1116 for more details.

When the code utilizes the address of an atomic variable, all the assignments of value or reference of value operation through the atomic variable address also need to be scaled by multiplying or dividing a step. Atomic variables also need to be scaled appropriately for all host-side memory operations.

Suggestions to Fix

For example, this original CUDA* code:

```
__device__ unsigned int a1;
__global__ void kernel(){
  ...
  unsigned int old_val = atomicInc(&a1, 0x7fffffff);
  ...
}
int main(){
  unsigned int result;
  unsigned int *d_addr;
  cudaGetSymbolAddress((void **)&d_addr, a1);
  cudaMemset(d_addr, 2, sizeof(unsigned int));
  kernel<<<1, 100>>>();
  ...
  cudaMemcpy(&result, d_addr, sizeof(unsigned int), cudaMemcpyDeviceToHost);
  ...
}
```

results in the following migrated SYCL* code:

```
dpct::global_memory<unsigned int, 0> a1;
void kernel(unsigned int &a1){
  ...
  /*
  DPCT1116:0: The atomicInc was migrated to dpct::atomic_fetch_add(&a1, 2) / 2 for performance,
and 2 is computed by (UINT_MAX + 1) / (‘0x7fffffff’ + 1). This migration requires the initial
value of ‘a1’ to be scaled by multiplying 2, and any usage of value of ‘a1’ outside atomic
function to be scaled by dividing 2.
  */
  unsigned int old_val = dpct::atomic_fetch_add<sycl::access::address_space::generic_space>(&a1,
2) / 2;
  ...
}
int main(){
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
  unsigned int result;
  unsigned int *d_addr;
  /*
  DPCT1117:3: There is atomicInc/Dec operation on 'a1' and value of 'a1' was scaled by migration
```

```
for performance, refer to DPCT1116. Using value of 'a1' through 'd_addr' should also be scaled
by dividing or multiplying 2. You may need to adjust the code.
  */
  *((void **)&d_addr) = a1.get_ptr();
  q_ct1.memset(d_addr, 2, sizeof(unsigned int)).wait();
  q_ct1.submit(
    [&](sycl::handler &cgh) {
      a1.init();
      auto a1_ptr_ct1 = a1.get_ptr();
      cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 100), sycl::range<3>(1, 1, 100)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel1(*a1_ptr_ct1);
        });
    });
  ...
  q_ct1.memcpy(&result, d_addr, sizeof(unsigned int)).wait();
  ...
}
```

which is rewritten to:

```
dpct::global_memory<unsigned int, 0> a1;
void kernel(unsigned int &a1){
  ...
  unsigned int old_val = dpct::atomic_fetch_add<sycl::access::address_space::generic_space>(&a1,
2) / 2;
  ...
}
int main(){
  dpct::device_ext &dev_ct1 = dpct::get_current_device();
  sycl::queue &q_ct1 = dev_ct1.default_queue();
  unsigned int result;
  unsigned int *d_addr;
  *((void **)&d_addr) = a1.get_ptr();
  q_ct1.memset(d_addr, 2 * 2, sizeof(unsigned int)).wait();
  q_ct1.submit(
    [&](sycl::handler &cgh) {
      a1.init();
      auto a1_ptr_ct1 = a1.get_ptr();
      cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 100), sycl::range<3>(1, 1, 100)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel1(*a1_ptr_ct1);
        });
    });
  ...
  q_ct1.memcpy(&result, d_addr, sizeof(unsigned int)).wait();
  result = result / 2;
  ...
}
```

DPCT1118

Message

SYCL group functions and algorithms must be encountered in converged control flow.

Detailed Help

SYCL* group functions and algorithms must be encountered in converged control flow for all work-items in a work-group. If a work-item completes the kernel and exits early without reaching a collective work-group operation like `barrier`, all the other work-items in the work-group reaching the collective operation will wait for the exited work-item.

Refer to How do I fix the issue of SYCL* code hanging due to work group level synchronization, such as a group barrier used in a conditional statement? for additional information.

Suggestions to Fix

For example, this original CUDA* code:

```
__global__ void kernel(float *data) {
  int tid = threadIdx.x;
  if (tid < 32) {
    if (data[tid] < data[tid + 32]) {
      data[tid] = data[tid + 32];
    }
    __syncthreads();
    ...
  }
}
```

results in the following migrated SYCL code:

```
void kernel(float *data, const sycl::nd_item<3> &item_ct1) {
  int tid = item_ct1.get_local_id(2);
  if (tid < 32) {
    if (data[tid] < data[tid + 32]) {
      data[tid] = data[tid + 32];
    }
    /*
    DPCT1118:0: SYCL group functions and algorithms must be encountered in converged control
flow. You should check this condition holds.
    */
    /*
    DPCT1065:1: Consider replacing sycl::nd_item::barrier() with
sycl::nd_item::barrier(sycl::access::fence_space::local_space) for better performance if there
is no access to global memory.
    */
    item_ct1.barrier();
  }
}
```

which is rewritten to:

```
void kernel(float *data, const sycl::nd_item<3> &item_ct1) {
  int tid = item_ct1.get_local_id(2);

  if (tid < 32) {
    if (data[tid] < data[tid + 32]) {
      data[tid] = data[tid + 32];
```

```
    }
  }
  item_ct1.barrier();
  ...
}
```

DPCT1119

Message

Migration of *<type or API>* is not supported, please try to remigrate with option: *<option name>*.

Detailed Help

The default behavior of Intel® DPC++ Compatibility Tool does not apply the experimental extension in the migration, you need to re-migrate with the recommend option in order to get the workable code.

Suggestions to Fix

For example, this original CUDA* code:

```
__device__ void testReduce(const cg::thread_block &cta) {
  cg::thread_block_tile<16> tile16 = cg::tiled_partition<16>(cta);
}
```

results in the following migrated SYCL* code:

```
void testReduce(const sycl::group<3> &cta, const sycl::nd_item<3> &item_ct1) {
  /*
  DPCT1007:0: Migration of tiled_partition is not supported.
  */
  /*
  DPCT1119:1: Migration of cooperative_groups::__v1::thread_block_tile<16> is not
  supported, please try to remigrate with option:
  --use-experimental-features=logical-group.
  */
  cg::thread_block_tile<16> tile16 = cg::tiled_partition<16>(cta);
}
```

which needs to be re-migrated with "–use-experimental-features=logical-group":

```
void testReduce(const sycl::group<3> &cta, const sycl::nd_item<3> &item_ct1) {
  dpct::experimental::logical_group tile16 =
      dpct::experimental::logical_group(item_ct1, item_ct1.get_group(), 16);
}
```

DPCT1120

Message

<variable name> is in device memory and passed into device functions as an extra argument. As *<variable name>* is a template variable here, then the declaration of an extra parameter in device functions, the variable referenced in command group, and device functions may be incorrect as a result. You may need to adjust the code.

Detailed Help

<variable name> is in device memory and passed into device functions as an extra argument. As *<variable name>* is a template variable here, then the declaration of an extra parameter in device functions, the variable referenced in command group, and device functions should be template variable also, the migration result may be incorrect. You may need to adjust the code.

Suggestions to Fix

For example, this original CUDA* code:

```
struct Foo {
  int x;
  __host__ __device__ Foo() {}
  __host__ __device__ ~Foo() {}
};

template <typename T> __constant__ T cmem;

__global__ void kernel() {
  ...
  int x = cmem<Foo>.x;
  ...
}

void foo() {
  ...
  kernel<<<1, 1>>>();
  ...
}
```

results in the following migrated SYCL* code:

```
struct Foo {
  int x;
  Foo() {}
  ~Foo() {}
};

/*
DPCT1120:0: cmem is in device memory and passed into device functions as an
extra arguments. As cmem is a template variable here, then the declaration of
an extra parameter in device functions, the variable referenced in command
group, and device functions may be incorrect as a result. You may need to
adjust the code.
```

```
*/
template <typename T> static dpct::constant_memory<T, 0> cmem;

void kernel(T cmem) {
  ...
  int x = cmem<Foo>.x;
  ...
}

void foo() {
  ...
  cmem.init();

  dpct::get_in_order_queue().submit([&](sycl::handler &cgh) {
    auto cmem_ptr_ct1 = cmem.get_ptr();

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel(*cmem_ptr_ct1);
        });
  });
  ...
}
```

which needs to be rewritten to:

```
struct Foo {
  int x;
  Foo() {}
  ~Foo() {}
};

template <typename T> static dpct::constant_memory<T, 0> cmem;

template <class T>
void kernel(T cmem) {
  ...
  int x = cmem.x;
  ...
}

void foo() {
  ...
  cmem<Foo>.init();

  dpct::get_in_order_queue().submit([&](sycl::handler &cgh) {
    auto cmem_ptr_ct1 = cmem<Foo>.get_ptr();

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel(*cmem_ptr_ct1);
        });
  });
  ...
}
```

DPCT1121

Message

Make sure that the *<argument>* which is used in the SYCL group function/algorithm is initialized.

Detailed Help

For SYCL group function/algorithm in SYCL* spec 4.17.2 and 4.17.3, the input scalar value must be initialized by all work-items in the group. This is necessary for compatibility with standard C++ rules and to ensure that compilers do not optimize too aggressively.

Suggestions to Fix

For example, this original CUDA* code:

```
__device__ void d(int var) {
  if (threadIdx.x == 0)
    var = 123;
  var = __shfl_sync(0xffffffff, var, 0);
}

__global__ void kernel() {
  int var;
  d(var);
}
```

results in the following migrated SYCL* code:

```
void d(int var, const sycl::nd_item<3> &item_ct1) {
  if (item_ct1.get_local_id(2) == 0)
    var = 123;
  /*
  DPCT1121:0: Make sure that the "var" which is used in the SYCL group
  function/algorithm is initialized.
  */
  var = dpct::select_from_sub_group(item_ct1.get_sub_group(), var, 0);
}
void kernel(const sycl::nd_item<3> &item_ct1) {
  int var;
  d(var, item_ct1);
}
```

which needs to be rewritten to:

```
void d(int var, const sycl::nd_item<3> &item_ct1) {
  if (item_ct1.get_local_id(2) == 0)
    var = 123;
  var = dpct::select_from_sub_group(item_ct1.get_sub_group(), var, 0);
}
void kernel(const sycl::nd_item<3> &item_ct1) {
```

```
  int var = 0;
  d(var, item_ct1);
}
```

DPCT1122

Message

‘<*original type*>’ is migrated to ‘<*migrated type*>’ in the template declaration; it may cause template function or class redefinition; please adjust the code.

Detailed Help

SYCL* vector types like sycl::long4 are defined by ‘using <type><elems> = vec<<storage-type>, <elems>>’ in SYCL2020 Vector types section 4.12.2, but sycl::longlong4 is not defined. In the migration, both ‘long4’ and ‘longlong4’ are migrated to sycl::long4, this may cause template function or class redefinition.

Suggestions to Fix

For example, this original CUDA* code:

```
template <typename T> struct C { T a; };
template <> struct C<longlong4> { longlong4 a; };
template <> struct C<long4> { long4 a; };
```

results in the following migrated SYCL code:

```
template <typename T> struct C { T a; };
/*
DPCT1122:0: 'longlong4' is migrated to 'sycl::long4' in the template
declaration; it may cause template function or class redefinition; please adjust
the code.
*/
template <> struct C<sycl::long4> { sycl::long4 a; };
template <> struct C<sycl::long4> { sycl::long4 a; };
```

which needs to be rewritten to:

```
template <typename T> struct C { T a; };
template <> struct C<sycl::long4> { sycl::long4 a; };
```

DPCT1123

Message

The kernel function pointer cannot be used in the device code. You need to call the kernel function with the correct argument(s) directly. According to the kernel function definition, adjusting the dimension of the sycl::nd_item may also be required

Detailed Help

Since SYCL* 2020 does not support calling a function pointer in a device function (SYCL 2020 Spec, 5.4. Language restrictions for device functions), the user needs to adjust the code to directly call the function pointed by that function pointer. Furthermore, the tool cannot do the analysis well for the dimension of sycl::nd_item (if option –assume-nd-range-dim=1 is used during migration) and the arguments, so the user may also need to adjust the related code.

Suggestions to Fix

For example, this original CUDA* code:

```
__global__ void kernel(int *d) {
  int gtid = blockIdx.x * blockDim.x + threadIdx.x;
  d[gtid] = gtid;
}

void foo(int *d) {
  void *kernel_array[100];
  kernel_array[10] = (void *)&kernel;
  void *args[1] = {&d};
  cudaLaunchKernel(kernel_array[10], dim3(16), dim3(16), args, 0, 0);
}
```

results in the following migrated SYCL code:

```
void kernel(int *d, const sycl::nd_item<3> &item_ct1) {
  int gtid = item_ct1.get_group(2) * item_ct1.get_local_range(2) +
             item_ct1.get_local_id(2);
  d[gtid] = gtid;
}

void foo(int *d) {
  sycl::device dev_ct1;
  sycl::queue q_ct1(dev_ct1,
                    sycl::property_list{sycl::property::queue::in_order()});
  void *kernel_array[100];
  kernel_array[10] = (void *)&kernel;
  void *args[1] = {&d};
  /*
  DPCT1123:0: The kernel function pointer cannot be used in the device code. You
  need to call the kernel function with the correct argument(s) directly.
  According to the kernel function definition, adjusting the dimension of the
  sycl::nd_item may also be required.
  */
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 16) * sycl::range<3>(1, 1, 16),
                        sycl::range<3>(1, 1, 16)),
      [=](sycl::nd_item<3> item_ct1) {
```

```
        (kernel_array[10])();
      });
}
```

which needs to be rewritten to:

```
void kernel(int *d, const sycl::nd_item<3> &item_ct1) {
  int gtid = item_ct1.get_group(2) * item_ct1.get_local_range(2) +
             item_ct1.get_local_id(2);
  d[gtid] = gtid;
}

void foo(int *d) {
  sycl::device dev_ct1;
  sycl::queue q_ct1(dev_ct1,
                    sycl::property_list{sycl::property::queue::in_order()});
  q_ct1.parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 16) * sycl::range<3>(1, 1, 16),
                        sycl::range<3>(1, 1, 16)),
      [=](sycl::nd_item<3> item_ct1) {
        kernel(d, item_ct1);
      });
}
```

DPCT1124

Message

<API name> is migrated to asynchronous memcpy API. While the origin API might be synchronous, it depends on the type of operand memory, so you may need to call wait() on event return by memcpy API to ensure synchronization behavior.

Detailed Help

Some CUDA* asynchronous API might be synchronous with respect to host. For example, asynchronous memcpy API might be synchronous in the following 3 cases: data transfers between device memory and pageable host memory, data transfers between host memory, and data transfers involving pinned memory.

Suggestions to Fix

For example, this original CUDA code:

```
cudaMemcpyAsync(host_dst, host_src, size, cudaMemcpyHostToHost);
```

results in the following migrated SYCL* code:

```
/*
DPCT1124:0: cudaMemcpyAsync is migrated to asynchronous memcpy API. While the
origin API might be synchronous, it depends on the type of operand memory, so
you may need to call wait() on event return by memcpy API to ensure
```

```
synchronization behavior.
*/
dpct::get_in_order_queue().memcpy(host_dst, host_src, size);
```

which needs to be rewritten to:

```
dpct::get_in_order_queue().memcpy(host_dst, host_src, size).wait();
```

DPCT1125

Message

The type "*<type name>*" defined in function "*<function name>*" is used as the parameter type in all functions in the call path from the corresponding sycl::handler::parallel_for() to the current function. You may need to adjust the definition location of the type.

Detailed Help

SYCL* 2020 does not support declaring a local memory variable in place. A local accessor needs to be declared in the command group scope and passed to all functions using the local memory. So, the definition location of the local memory variable type needs to be changed. Note: The DPC++ compiler extension sycl_ext_oneapi_local_memory (https://github.com/intel/llvm/blob/sycl/sycl/doc/extensions/supported/sycl_ext_oneapi_local_memory.asciidoc) supports declaring a local memory variable in place, but only in the kernel function scope.

Suggestions to Fix

For example, this original CUDA* code:

```
template <typename T> struct kernel_type_t {
  using Type = T;
};

template <typename T> __global__ void device(int a, int b) {
  using Tk = typename kernel_type_t<T>::Type;
  __shared__ Tk mem[256];
  ...
}

template <typename T> __global__ void kernel(int a, int b) { device<T>(a, b); }

template <typename T> void foo() {
  int i;
  kernel<T><<<1, 1>>>(i, i);
}
```

results in the following migrated SYCL code:

```
template <typename T> struct kernel_type_t {
  using Type = T;
};
```

```
/*
DPCT1125:1: The type "Tk" defined in function "device" is used as the parameter
type in all functions in the call path from the corresponding
sycl::handler::parallel_for() to the current function. You may need to adjust
the definition location of the type.
*/
template <typename T> void device(int a, int b, Tk *mem) {
  using Tk = typename kernel_type_t<T>::Type;

  ...
}

/*
DPCT1125:2: The type "Tk" defined in function "device" is used as the parameter
type in all functions in the call path from the corresponding
sycl::handler::parallel_for() to the current function. You may need to adjust
the definition location of the type.
*/
template <typename T>
void kernel(int a, int b, Tk *mem) {
  device<T>(a, b, mem);
}

template <typename T> void foo() {
  int i;
  /*
  DPCT1126:0: The type "Tk" defined in function "device" is used as the
  parameter type in all functions in the call path from the
  sycl::handler::parallel_for() to the function "device". You may need to adjust
  the definition location of the type.
  */
  dpct::get_in_order_queue().submit([&](sycl::handler &cgh) {
    sycl::local_accessor<Tk, 1> mem_acc_ct1(sycl::range<1>(256), cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel<T>(
              i, i,
              mem_acc_ct1.template get_multi_ptr<sycl::access::decorated::no>()
                  .get());
        });
  });
}
```

which needs to be rewritten to:

```
template <typename T> struct kernel_type_t {
  using Type = T;
};

template <typename T> void device(int a, int b, typename kernel_type_t<T>::Type *mem) {
  ...
}

template <typename T>
void kernel(int a, int b, typename kernel_type_t<T>::Type *mem) {
  device<T>(a, b, mem);
```

```
}

template <typename T> void foo() {
  int i;
  using Tk = typename kernel_type_t<T>::Type;
  dpct::get_in_order_queue().submit([&](sycl::handler &cgh) {
    sycl::local_accessor<Tk, 1> mem_acc_ct1(sycl::range<1>(256), cgh);

    cgh.parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
          kernel<T>(
              i, i,
              mem_acc_ct1.template get_multi_ptr<sycl::access::decorated::no>().get());
        });
  });
}
```

DPCT1126

Message

The type "*<type name>*" defined in function "*<function name>*" is used as the parameter type in all functions in the call path from the sycl::handler::parallel_for() to the function "*<function name>*". You may need to adjust the definition location of the type.

Detailed Help

See DPCT1125.

Suggestions to Fix

See DPCT1125.

DPCT1127

Message

The constant compile-time initialization for device_global is supported when compiling with C++20. You may need to adjust the compile commands.

Detailed Help

The device_global extension introduces device scoped memory allocations into SYCL* that can be accessed within a kernel, using syntax similar to C++ global variables. Constant compile-time initialization for device_global is supported when compiling with C++20. Ref: https://github.com/intel/llvm/blob/sycl/sycl/doc/extensions/experimental/sycl_ext_oneapi_device_global.asciidoc.

Suggestions to Fix

For example, this original CUDA* code:

```
// test.cu
__constant__ int var = 0;
__global__ void kernel(int* ptr) {
  *ptr = var;
}
```

results in the following migrated SYCL code:

```
// test.dp.cpp
/*
DPCT1127:0: The constant compile-time initialization for device_global is
supported when compiling with C++20. You may need to adjust the compile
commands.
*/
static sycl::ext::oneapi::experimental::device_global<const int> var{0};
void kernel(int* ptr) {
  *ptr = var.get();
}
```

The compile commands need to be rewritten to:

```
icpx -fsycl -std=c++20 test.dp.cpp # Linux
icx-cl -fsycl -Qstd=c++20 test.dp.cpp # Windows
```

DPCT1128

Message

The type "<*type name*>" is not device copyable for <*reason*>. It is used in the SYCL kernel, please rewrite the code.

Detailed Help

SYCL* 2020 needs the type of argument passed to the kernel function to be device copyable (Spec 3.13.1 device copyable), while CUDA* just does memory copy for all those arguments directly without emitting any error/warning. If a type does not meet the requirements of device copyable, the tool will try to add the specialization of the sycl::is_device_copyable for this type, as well as warn the user. The warning will specify which part of that type does not meet the requirements.

Suggestions to Fix

For example, this original CUDA code:

```
template <class T1, class T2> struct UserStruct {
  UserStruct() {}
  UserStruct(UserStruct const &other) : a(other.a) {}
  int a;
};

template <class V1, class V2> __global__ void k(UserStruct<V1, V2>) {}

void foo() {
  UserStruct<float, int> us;
  k<<<1, 1>>>(us);
}
```

results in the following migrated SYCL code:

```
/*
DPCT1128:0: The type "UserStruct<float, int>" is not device copyable for copy
constructor breaking the device copyable requirement. It is used in the SYCL
kernel, please rewrite the code.
*/
template <class T1, class T2> struct UserStruct {
  UserStruct() {}
  UserStruct(UserStruct const &other) : a(other.a) {}
  int a;
};
template <class T1, class T2>
struct sycl::is_device_copyable<UserStruct<T1, T2>> : std::true_type {};

template <class V1, class V2> void k(UserStruct<V1, V2>) {}

void foo() {
  UserStruct<float, int> us;
  /*
  DPCT1129:1: The type "UserStruct<float, int>" is used in the SYCL kernel, but
  it is not device copyable. The sycl::is_device_copyable specialization has
  been added for this type. Please review the code.
  */
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
        k(us);
      });
}
```

which needs to be rewritten to:

```
template <class T1, class T2> struct UserStruct {
  UserStruct() {}
  UserStruct(UserStruct const &other) : a(other.a) {}
  int a;
};
template <class T1, class T2>
struct sycl::is_device_copyable<UserStruct<T1, T2>> : std::true_type {};

template <class V1, class V2> void k(UserStruct<V1, V2>) {}

void foo() {
```

```
  UserStruct<float, int> us;
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
        k(us);
      });
}
```

```
template <class T1, class T2> struct UserStruct {
  UserStruct() {}
  int a;
};

template <class V1, class V2> void k(UserStruct<V1, V2>) {}

void foo() {
  UserStruct<float, int> us;
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
      [=](sycl::nd_item<3> item_ct1) {
        k(us);
      });
}
```

DPCT1129

Message

The type "*<type name>*" is used in the SYCL kernel, but it is not device copyable. The sycl::is_device_copyable specialization has been added for this type. Please review the code.

Detailed Help

See DPCT1128.

Suggestions to Fix

See DPCT1128.

DPCT1130

Message

SYCL 2020 standard does not support dynamic parallelism (launching kernel in device code). Please rewrite the code.

Detailed Help

SYCL* does not support launching kernel in device code. The user needs to merge the parent kernel and child kernel together.

Suggestions to Fix

For example, this original CUDA* code:

```
__global__ void childKernel() {
      ...
}
__global__ void parentKernel() {
      ...
      childKernel<<<4, 4>>>();
      ...
}
void foo() {
      ...
      parentKernel<<<8, 8>>>();
      ...
}
```

results in the following migrated SYCL code:

```
void childKernel() {
  ...
}
void parentKernel() {
  ...
  /*
  DPCT1130:0: SYCL 2020 standard does not support dynamic parallelism (launching
  kernel in device code). Please rewrite the code.
  */
  childKernel<<<4, 4>>>();
  ...
}
void foo() {
  ...
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 8) * sycl::range<3>(1, 1, 8),
                        sycl::range<3>(1, 1, 8)),
      [=](sycl::nd_item<3> item_ct1) {
        parentKernel();
      });
  ...
}
```

which is rewritten to:

```
void childKernel() {
  ...
}
void parentKernel() {
```

```
  ...
  childKernel(); // call childKernel() as a device function, need to adjust the work
  for each work item.
  ...
}
void foo() {
  ...
  dpct::get_in_order_queue().parallel_for(
      sycl::nd_range<3>(sycl::range<3>(1, 1, placeholder /*Adjust the global range
      based on the thread model between parentKernel and childKernel*/),
                        sycl::range<3>(1, 1, placeholder /*Adjust the local range
                        based on the thread model between parentKernel and
                        childKernel */)),
      [=](sycl::nd_item<3> item_ct1) {
        parentKernel();
      });
  ...
}
```

DPCT1131

Message

The migration of "*<API name>*" is not currently supported with SYCLcompat. Please adjust the code manually.

Detailed Help

When using syclcompat header-only library to assist the migration, this API migration is not supported.

Suggestions to Fix

This kind of code requires manual migration to SYCL*. Please refer to the migration result without specifying "–use-syclcompat".

DPCT1132

Message

SYCL 2020 does not support accessing the *<property name>* for the kernel. The API is replaced with member variable "*<variable name>*". Please set the appropriate value for "*<variable name>*".

Detailed Help

There is limited support for querying the kernel attribute such as shared memory or number of registers used by a kernel in SYCL* 2020. The tool migrates the following attributes to member variables of a class defined in the DPCT namespace helper function, in order to ensure the compilability of migrated code.

- Shared-memory size per-block required by the kernel
- Local memory size used by each thread in the kernel
- Constant memory size required by the kernel
- Number of registers used by each thread in the kernel

It is recommend to use an appropriate value to replace the member variable access, provided it is statically available. Otherwise, consider removing the code along with the part that depends on it.

Suggestions to Fix

For example, this original CUDA* code:

```
void foo(CUfunction f) {
  int smsize;
  int lcsize;
  int cnsize;
  int nreg;

  cuFuncGetAttribute(&smsize, CU_FUNC_ATTRIBUTE_SHARED_SIZE_BYTES, f);

  cuFuncGetAttribute(&lcsize, CU_FUNC_ATTRIBUTE_LOCAL_SIZE_BYTES, f);

  cuFuncGetAttribute(&cnsize, CU_FUNC_ATTRIBUTE_CONST_SIZE_BYTES, f);

  cuFuncGetAttribute(&nreg, CU_FUNC_ATTRIBUTE_NUM_REGS, f);
}
```

results in the following migrated SYCL code:

```
void foo(dpct::kernel_function f) {
  int smsize;
  int lcsize;
  int cnsize;
  int nreg;
  /*
  DPCT1132:0: SYCL 2020 does not support accessing the statically allocated
  shared memory for the kernel. The API is replaced with member variable
  "shared_size_bytes". Please set the appropriate value for "shared_size_bytes".
  */
  smsize = dpct::get_kernel_function_info(f)
               .shared_size_bytes /* statically allocated shared memory per
                                     work-group in bytes */
      ;

  /*
  DPCT1132:1: SYCL 2020 does not support accessing the local memory for the
  kernel. The API is replaced with member variable "local_size_bytes". Please
  set the appropriate value for "local_size_bytes".
  */
  lcsize = dpct::get_kernel_function_info(f)
               .local_size_bytes /* local memory per work-item in bytes */;

  /*
```

```
  DPCT1132:2: SYCL 2020 does not support accessing the memory size of
  user-defined constants for the kernel. The API is replaced with member
  variable "const_size_bytes". Please set the appropriate value for
  "const_size_bytes".
  */
  cnsize =
      dpct::get_kernel_function_info(f)
          .const_size_bytes /* user-defined constant kernel memory in bytes */;

  /*
  DPCT1132:3: SYCL 2020 does not support accessing the required number of
  registers for the kernel. The API is replaced with member variable
  "num_regs". Please set the appropriate value for "num_regs".
  */
  nreg = dpct::get_kernel_function_info(f)
             .num_regs /* number of registers for each thread */;
}
```

which is rewritten to:

```
void foo(dpct::kernel_function f) {
  int smsize;
  int lcsize;
  int cnsize;
  int nreg;

  smsize = 1024 /* statically allocated shared memory per-group in bytes according
  to kernel function f */;

  lcsize = 1024 /* local memory per work-item in bytes according to kernel function
  f */;

  cnsize = 1024 /* user-defined constant kernel memory in bytes according to kernel
  function f */;

  nreg = 4 /* number of registers for each thread according to kernel function f */;
}
```

DPCT2001

Message

You can link with more libraries by adding them here.

Detailed Help

When the build script **Makefile.dpct** is generated, the warning message is inserted into the build script **Makefile.dpct**, as shown in the following example:

```
#content of Makefile.dpct
...
warning: #DPCT2001:228: You can link with more libraries by adding them here.
LIB :=
...
```

Suggestions to Fix

For example, if an application needs to link the OpenCV libraries in link time, fix the **LIB** variable as follows:

```
LIB :=  -lopencv_core -lopencv_imgcodecs
```

DPCT3000

Message

Migration of syntax "*<CMake CUDA* specific command>*" is not supported. You may need to adjust the code.

Detailed Help

Migration of syntax "*<CMake CUDA specific command>*" is not supported. You may need to adjust the code.

Suggestions to Fix

If there is straightforward mapping available on the SYCL* side, use the mapping instead in the migrated CMake script; If not, you need to implement a helper function in dpct.cmake, which is part of the migrated project, and use this helper function as the mapping in the migrated CMake script.

DPCT3001

Message

"<CMake common command>" is used to generate an output file by performing transformation on input file. You may need to update the name of the input and output file, as well as migrate the input file if the input file contains CUDA syntax.

Detailed Help

"<CMake common command>" is used to generate an output file by performing transformation on input file. You may need to update the name of the input and output file, as well as migrate the input file if the input file contains CUDA syntax.

Suggestions to Fix

For example, the CMake command configure_file(<input> <output> options) with the "@ONLY" option replaces variables in <input> file that are explicitly marked with "@VAR@" format (see "@FOO_VERSION@" in the example code) with the value of VAR defined in CMake script(see "FOO_VERSION" in the CMake script), and then save out as <output> file.

For example, this original CUDA* code:

```
// src/bar.in.cu
__global__ void hello_kernel() {}

int main() {
    std::cout << "FOO version: @FOO_VERSION@" << std::endl;
    hello_kernel<<<1, 1>>>();
    cudaDeviceSynchronize();
    return 0;
}
```

And the CMake script for the original CUDA* code:

```
# src/CMakeLists.txt
set(FOO_VERSION 1.00)

configure_file(
    ${CMAKE_SOURCE_DIR}/bar.in.cu
    ${CMAKE_BINARY_DIR}/bar.cu
    @ONLY
)
```

result in the following migrated SYCL codes:

```
// build/bar.dp.cpp
void hello_kernel() {
}

int main() {
    std::cout << "FOO version: 1.00" << std::endl;  // Example replacement
    dpct::get_in_order_queue().parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
            hello_kernel();
        });
    dpct::get_current_device().queues_wait_and_throw();
    return 0;
}
```

```
# CMakeLists.txt
set(FOO_VERSION 1.00)
DPCT3001:1: "configure_file" is used to generate an output file by performing
transformation on input file. You may need to update the name of the input and
output file, as well as migrate the input file if the input file contains CUDA syntax
configure_file(
    ${CMAKE_SOURCE_DIR}/bar.in.dp.cpp
    ${CMAKE_BINARY_DIR}/bar.dp.cpp
    @ONLY
)
```

which need to be rewritten to:

```
// src/bar.in.dp.cpp
void hello_kernel() {
}

int main() {
    std::cout << "FOO version: @FOO_VERSION@" << std::endl;
    dpct::get_in_order_queue().parallel_for(
        sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)),
        [=](sycl::nd_item<3> item_ct1) {
            hello_kernel();
        });
    dpct::get_current_device().queues_wait_and_throw();
    return 0;
}
```

and:

```
# src/CMakeLists.txt
set(FOO_VERSION 1.00)
configure_file(
    ${CMAKE_SOURCE_DIR}/bar.in.dp.cpp
    ${CMAKE_BINARY_DIR}/bar.dp.cpp
    @ONLY
)
```

DPCT3002

Message

"<Library specific to CUDA>" is removed because it is not supported for use alongside option "-fsycl" of SYCL compiler. You may need to check the build options.

Detailed Help

"<Library specific to CUDA>" is removed because it is not supported for use alongside option "-fsycl" of SYCL compiler. You may need to check the build options.

Suggestions to Fix

For example, this original CUDA* code:

```
target_link_options(nvcv_util_compat
PUBLIC
    -static-libstdc++
    -static-libgcc
    -Wl,--wrap=__libc_start_main
    -Wl,-u__cxa_thread_atexit_impl
)
```

results in the following migrated CMake script:

```
target_link_options(nvcv_util_compat
PUBLIC
    # DPCT3002:1: "-static-libstdc++" is removed because it is not supported for use
    alongside option "-fsycl" of SYCL compiler. You may need to check the build
    options.
    -static-libgcc
    -Wl,--wrap=__libc_start_main
    -Wl,-u__cxa_thread_atexit_impl
)
```

which is rewritten to:

```
target_link_options(nvcv_util_compat
PUBLIC
    -static-libgcc
    -Wl,--wrap=__libc_start_main
    -Wl,-u__cxa_thread_atexit_impl
)
```

DPCT Namespace Reference

The DPCT namespace provides helper functions and macros to assist the migration of input source code. DPCT namespace helper functions are implemented in header files and are available in the include folder (include/dpct) of the Intel® DPC++ Compatibility Tool install path. These header files are intended to become part of your new SYCL* code generated by Intel® DPC++ Compatibility Tool. You can copy these header files if needed and include them with the generated code.

There are three namespaces defined:

Namespace	Description
`dpct::`	This namespace is used in the code generated by Intel® DPC++ Compatibility Tool.
`dpct::detail::`	This namespace is an internal implementation and should not be used in SYCL source files directly, as the API and implementations inside the `dpct::detail::` namespace can be changed at any time without notice.
`dpct::deprecated::`	This namespace is for functionality that has been introduced for compatibility purposes, but relies on deprecated C++ features, that have been removed or will be removed in the future standards. Use of deprecated functionality in the `dpct::deprecated::` namespace should be avoided.

NOTE After migration, it is recommended that you continue development using SYCL and C++ standard APIs, rather than continue development based on DPCT namespace helper functions and macros.

The following sections describe different domains of helper functions and macros in more detail.

Macros

The following table lists the macros used to control the behavior of the helper functions:

Macro	Description
DPCT_USM_LEVEL_NONE	If the --usm-level=none option of Intel® DPC++ Compatibility Tool is used, then DPCT_USM_LEVEL_NONE is defined before DPCT namespace helper header files are included into migrated code. By default, this macro is not defined. If DPCT_USM_LEVEL_NONE is defined, corresponding helper functions guarded by this macro are active, and the resulting migrated code will use buffer-based memory functions instead of USM-based memory functions.
DPCT_NAMED_LAMBDA	If the --sycl-named-lambda option of Intel® DPC ++ Compatibility Tool is used, then DPCT_NAMED_LAMBDA is defined before DPCT namespace helper header files are included in migrated code. By default, this macro is not defined. If DPCT_NAMED_LAMBDA is defined, corresponding helper code guarded by this macro is active and the resulting migrated code is using it.
DPCT_COMPATIBILITY_TEMP	DPCT_COMPATIBILITY_TEMP is defined as an integer number in helper header files, and is inserted into migrated code by Intel® DPC++ Compatibility Tool. You can replace this macro in the generated code or change the value of DPCT_COMPATIBILITY_TEMP according to your needs.

NOTE Do not manually define these macros. They are automatically defined by Intel® DPC++ Compatibility Tool according to the options specified in the dpct command line.

Atomic Helper Functions

The atomic helper functions are available in the atomic.hpp header file. Atomic helper functions are wrappers around standard SYCL atomic operations.

Device Helper Functions

The device helper functions are available in the device.hpp header file.

The following classes are defined in the device.hpp header file:

Class	Description
device_info	device_info is used to present information enumerated by SYCL device information descriptors.
device_ext	device_ext is a subclass of SYCL device class cl::sycl::device.

Class	Description
	`device_ext` adds a default SYCL queue associated with the device on top of SYCL device, along with interface to query device information, and initializes `device_info` class.
`dev_mgr`	`dev_mgr` is a singleton, intended to manage all SYCL-capable devices available on the system (for example CPU or GPU). `dev_mgr` selects one of the available devices as current device, according to the behavior of SYCL default device selector. Display the current device using `current_device()`, and change the current device using `select_device()`.

Find more details in the `device.hpp` file.

The following table lists the device helper functions:

Function	Description
`get_current_device()`	Returns the current device in the device manager.
`get_default_queue()`	Returns the default queue associated with the current device in the device manager.
`get_device(unsigned int id)`	Returns the `device_ext`object by integer id.
`cpu_device`	Returns the `device_ext` object associated with the SYCL CPU device.
`get_default_context()`	Returns the SYCL context of the default queue associated with the current device in device manager.

Memory Helper Functions

The following table lists SYCL buffer-based memory model helper functions:

Function	Description
Device Memory Manager (`mem_mgr`)	Device memory manager is a singleton, intended to assist with migration of memory API calls if a SYCL buffer-based memory model is used. It provides API to allocate and free host virtual address space bound to `cl::sycl::buffer`, and transform between the host virtual address and `cl::sycl::buffer`. By default, the `mem_mgr` maintains 128GB of virtual memory space. You can change virtual memory space as required by changing the value of `mapped_region_size` in the `memory.hpp` file.

The following table lists helper functions to get the SYCL buffer from host virtual address:

Function	Description
`get_buffer_and_offset(const void *ptr)`	Where `ptr`is the host virtual address. This function returns the `cl::sycl::buffer` bound to the host virtual address, and the offset of the host virtual address in the `cl::sycl::buffer`.
`get_buffer(const void *ptr)`	Where `ptr`is the host virtual address. Returns the `sycl::buffer` bound to the host virtual address.

General Helper Functions

General memory helper functions are used to assist the migration of memory API calls, like memory allocation, memory copy, and memory set, for both USM and buffer-based memory model. The following helper functions are available:

- `dpct_malloc()`
- `dpct_free()`
- `dpct_memcpy()`
- `async_dpct_memcpy()`
- `dpct_memset()`
- `async_dpct_memset()`

Find more details in the `memory.hpp` file.

Helper Functions Assisting Migration of Device Memory Variables

To assist with migration of variables declared with global and constant device memory attributes, the helper functions provide the following template classes:

Function	Description
`global_memory`	Represents a SYCL global memory variable.
`constant_memory`	Represents a SYCL constant memory variable.
`shared_memory`	Represents memory that can be accessed by host and device, in the same way as SYCL USM shared memory.

The following table lists main member functions of memory classes:

Function	Description
`get_prt()`	Returns memory pointer of the memory object. It is a virtual pointer when buffer-based memory model is used, and it is a device pointer when a USM memory model is used.
`get_size()`	Returns the size of a memory object.
`get_access()`	Returns the accessor to the memory object. If a USM memory model is used, this function returns a `dpct_accessor_t` object for the device memory object. `dpct_accessor_t` is a helper type, which contains the data pointer, data dimension information, and provides a method to access the

Function	Description
	memory. If a buffer-based memory model is used, this function returns `cl::sycl::accessor` for the device memory object.
`init()`	Allocates memory after a SYCL device is selected, and initializes device memory if it has initial value. This API should be called after the SYCL device is selected.

Kernel Helper Functions

Kernel helper functions provide a structure `kernel_function_info` to keep SYCL kernel information, and provide a utility function `get_kernel_function_info()` to get the kernel information.

Image Helper Functions

The image helper functions provide class wrappers to assist the migration of texture API. Find more details on image helper functions in the `image.hpp` file.

Wrapper	Description
Template class `image_wrapper`	The wrapper of `cl::sycl::image`. It provides methods to attach/detach 1D/2D/3D data to/from `cl::sycl::image` and method `get_access()` to return the SYCL image accessor.
Class `image_accessor_ext`	Wraps SYCL sampler and SYCL image accessor together. Provides the `read()` method to read data from the accessor.
Class `image_data`	Represents the data used to create a `cl::sycl::image` object. Supports the following data types: • Matrix for 2D or 3D matrix data (see `image_matrix` class) • Linear for 1D linear array • Pitch for pitched 2D or 3D memory data (`pitched_data` class)
Class `image_matrix`	Represents 2D/3D matrix data for an image. It contains the image channel information, dimensions, size of each dimension, and the raw data pointer.
Class `pitched_data`	Represents the 2D/3D pitched data for an image.
Struct `image_channel`	Keeps image channel information; including order, data width, and type information.
Class `image_info`	Image sampling information; including addressing mode, filtering mode, and normalization mode.
`image_wrapper` operations API: `create_image_wrapper()`	Creates an object of the helper image class according to the parameters provided.

Intel® oneAPI DPC++ Library Specific Utility Helper functions

The helper functions provided by the header files included in `dpl_utils.hpp` include algorithm functions, memory functions, iterators, and a container for managing data on a device.

Algorithm Helper Functions

The algorithm helper functions are available in the `dpl_extras/algorithm.h` header file. Algorithm helper functions are wrappers around Intel® oneAPI DPC++ Library (oneDPL) algorithm functions.

Mask-predicated Algorithm Functions

Mask-predicated algorithm functions have parameter lists like the algorithms with the same name in the C++ standard library. These functions accept an additional iterator pointing to a set of values that indicate whether the corresponding elements of the other inputs to the function should be processed by the algorithm. The predicate provided to these functions is applied to a value in the mask, rather than the element of the input data, and if the predicate returns true the corresponding element of the input data is manipulated by the algorithm.

The mask-predicated algorithm functions in the `dpl_extras/algorithm.h` header file are:

- `copy_if`
- `partition`
- `partition_copy`
- `remove_if`
- `remove_copy_if`
- `replace_if`
- `replace_copy_if`
- `stable_partition_copy`
- `stable_partition`
- `transform_if`

Key-value Pair Algorithm Functions

Key-value pair algorithm functions have parameter lists like the algorithms with the same name in the C++ standard library. The functions accept two sets of input data instead of one: the first representing a set of keys and the second representing the set of values associated with each key. These algorithm functions create `zip_iterators` and invoke the standard algorithm provided in oneDPL so that the key-value pairs are processed as single data elements.

The key-value pair algorithm functions in the `dpl_extras/algorithm.h` header file are:

- `unique`
- `unique_copy`
- `merge`
- `sort`
- `stable_sort`
- `set_difference`
- `set_intersection`
- `set_symmetric_difference`
- `set_union`

Additional Algorithm Functions

Function	Description
`partition_point`	Implementation of the C++ standard library algorithm that accepts an execution policy, enabling parallel execution.

Function	Description
`transform_if`	Applies a unary operation to every element in the input for which the predicate provided returns true. A mask-predicated version of the function is also provided.
`iota`	Extends the C++ standard library algorithm by accepting a step in addition to the input iterators and initial value. The step is used to compute the next value in the sequence generated.
`inner_product`	Implementation of the C++ standard library algorithm that accepts an execution policy, enabling parallel execution. The binary operation provided for reduction must be associative and commutative.
`for_each_index`	Assigns each element in the input range provided the result of the unary operator invoked with the element's index in the input range.

Memory Helper Functions

The `dpl_extras/memory.h` header file provides classes and functions for working with device memory.

The following table lists memory helper classes:

Class	Description
`device_pointer`	Pointer type that points to memory stored on a device.
`device_iterator`	Iterator type returned by `device_vector` begin and end methods.
`device_reference`	Reference type returned by `device_vector` element access operator.

The `device_vector` implementation that uses the `device_iterator` and `device_reference` classes is defined in `dpl_extras/vector.h`.

The following table list memory helper functions:

Function	Description
`malloc_device`	Allocates memory for use on a device.
`free_device`	Deallocates memory allocated with `malloc_device`.
`device_new`	Allocates memory for use on a device large enough to hold the requested number of elements of the type specified.
`device_delete`	Deallocates memory allocated using `device_new`.
`get_device_pointer`	Constructs a `device_pointer` from a raw pointer.

Function	Description
`get_raw_pointer`	Returns a raw pointer from a `device_pointer` instance.

oneAPI DPC++ Library Specific Utility Header Files

The following table lists utility header files specific to the oneAPI DPC++ Library:

Header File	Description
`functional.h`	Contains the implementation of the function objects used in the definitions of the algorithm functions described above.
`iterator.h`	Provides an implementation of `constant_iterator`, an iterator that is constructed with a value and when dereferenced will only return that value.
`vector.h`	Provides an implementation of `device_vector`, a container for managing data elements stored in device memory.

Utility Helper Functions

The following table lists header files that define utility functions:

Header File	Description
`util.hpp`	Common utility helper functions.
`blas_utils.hpp`	BLAS-specific utility helper functions.

CUDA* and SYCL* Programming Model Comparison

This section compares the CUDA* and SYCL* programming models and shows how to map concepts and APIs from CUDA to SYCL.

Execution Model

Kernel Function

In CUDA*, a kernel function is defined using the `__global__` declaration specifier and is executed concurrently across multiple threads on the GPU. Functions that are called by a CUDA kernel must be qualified with the `__device__` specifier. The `__host__` declaration specifier is used to qualify functions that can be called from host code running on the CPU.

In contrast, a SYCL* kernel is a function that is executed on SYCL-capable devices, such as CPUs, GPUs, or FPGAs. These kernels are launched from host code and are executed concurrently on SYCL devices. Unlike CUDA, SYCL kernel functions do not require special declaration specifiers and are defined using standard C++ syntax.

The following table shows CUDA and SYCL equivalencies for defining kernel functions:

CUDA	SYCL
`__global__ void foo_kernel() {}`	`void foo_kernel() {}`
`__device__ void foo_device() {}`	`void foo_device() {}`
`__host__ void foo_host() {}`	`void foo_host() {}`
`__host__ __device__ void foo_host_device() {}`	`void foo_host_device() {}`

If the macro `__CUDA_ARCH__` is used to differentiate the code path in a CUDA `__host__ __device__` function, you can use macro `__SYCL_DEVICE_ONLY__` in SYCL to achieve similar functionality.

The following table shows CUDA and SYCL equivalencies specifying device functions:

CUDA	SYCL

`__host__ __device__ int foo(int i) {`
`#ifdef __CUDA_ARCH__`
`  return i + 1;`
`#else`
`  return 0;`
`#endif`
`}`
`__global__ void kernel() {`
`  foo();`
`}`
`int main() {`
`  return foo();`
`}`
	`int foo(int i) {`
	`#ifdef __SYCL_DEVICE_ONLY__`
	`  return i + 1;`
	`#else`
	`  return 0;`
	`#endif`
	`}`
	`void kernel() {`
	`  foo();`
	`}`
	
	`int main() {`
	`  return foo();`
	`}`

Execution Hierarchy

In both CUDA and SYCL programming models, the kernel execution instances are organized hierarchically to exploit parallelism effectively. In CUDA, these instances are called threads; in SYCL, they are referred to as work-items. CUDA threads can be organized into blocks, which in turn can be organized into grids. SYCL work-items can be organized into work-groups, which in turn can be organized into ND-ranges.

From a hardware perspective, when a CUDA kernel is executed on a GPU, the Streaming Multiprocessors (SMs) create, manage, schedule, and execute threads in groups of 32 threads, known as warps. In comparison, in SYCL a sub-group represents a collection of related work-items within a work-group that execute concurrently.

To migrate CUDA code to SYCL code, the CUDA execution hierarchy can be mapped to the SYCL hierarchy as shown in the following table:

CUDA	SYCL
thread	work-item
warp	sub-group

CUDA	SYCL
block	work-group
grid	ND-range

Thread Indexing

As previously discussed, CUDA threads and SYCL work-items are organized in a hierarchical manner.

CUDA contains built-in variables to support threads:

- Thread ID: `threadIdx.x/y/z`
- Block ID: `blockIdx.x/y/z`
- Block dimensions: `blockDim.x/y/z`
- Grid dimensions: `gridDim.x/y/z`

SYCL contains equivalent built-in variables:

- Thread ID: `sycl::nd_item.get_local_id(0/1/2)`
- Work-group ID: `sycl::nd_item.get_group(0/1/2)`
- Work-group dimensions: `sycl::nd_item.get_local_range().get(0/1/2)`
- ND-range dimensions: `sycl::nd_item.get_group_range(0/1/2)`

According to the execution space index linearization sections of the CUDA C++ programing Guide and the SYCL 2020 Specification, given a block or work-group with shape *(dx, dy, dz)* and an element with id *(x, y, z)*, then the index is *x + y * Dx + z * Dx * Dy* for CUDA, and *z + y * Dz + x * Dz * Dy* for SYCL. This discrepancy is due in part to SYCL's better alignment with C++'s multidimensional indexing. Due to the difference in index computation methods, the CUDA execution space right-most dimension *z* is mapped to the SYCL left-most dimension *x*, as shown in the following table:

CUDA	SYCL
`gridDim.x/y/z`	`sycl::nd_item.get_group_range(2/1/0)`
`blockIdx.x/y/z`	`sycl::nd_item.get_group(2/1/0)`
`blockDim.x/y/z`	`sycl::nd_item.get_local_range().get(2/1/0)`
`threadIdx.x/y/z`	`sycl::nd_item.get_local_id(2/1/0)`
`warpsSize`	`sycl::nd_item.get_sub_group().get_local_range().get(0)`

Kernel Launch

CUDA uses the `<<<...>>>` execution configuration syntax and the `dim3` type to specify the dimensions and sizes of grids and blocks. The function call operator, in conjunction with the execution configuration syntax, is used to submit a kernel function to a stream for execution. When a kernel is submitted, an index space is defined, with each thread and block receiving a unique thread ID and block ID. These IDs are used to compute the index within the index space, and they are accessible within the kernel through built-in variables.

SYCL uses the `parallel_for` member function of `sycl::queue` and `sycl::range` to specify the dimensions and sizes of ND-ranges and work-groups. Users can also apply the SYCL kernel attribute `[[sycl::reqd_sub_group_size(dim)]]` to indicate that a kernel must be compiled and executed with the specified sub-group size. Each device supports only certain sub-group sizes, as defined by `info::device::sub_group_sizes`.

The following table shows the original CUDA code for a kernel launch example, migrated to SYCL:

Original CUDA Code	Migrated SYCL Code

`__global__ void foo() {`
`  int a = threadIdx.x;`
`}`
 	
`int main() {`
`  dim3 size_1(100, 200, 300);`
`  dim3 size_2(5, 10, 20);`
 	
`  foo<<<size_1, size_2>>>();`
`}`
	`void foo(sycl::nd_item<3> item) {`
	`  int a = item.get_local_id(2);`
	`}`
	
	`int main() {`
	`  sycl::queue q;`
	`  sycl::range<3> size_1(300, 200, 100);`
	`  sycl::range<3> size_2(20, 10, 5);`
	
	`  q.parallel_for(`
	`  sycl::nd_range<3>(size_1 * size_2,`
	`size_2),`
	`  [=](sycl::nd_item<3> item)`
	`[[sycl::reqd_sub_group_size(32)]] {`
	`  foo(item);`
	`  });`
	`}`

Make note of the following details in the migrated SYCL code:

- In the constructor of `sycl::nd_range`, the first parameter **global size** is a work-item instead of a work-group. Thus **global size** should be the product of `size_1` and `size_2` to align with CUDA.
- The Thread Indexing section describes that the CUDA execution space right-most dimension *z* should be mapped to the SYCL left-most dimension *x*. Thus in this example, the size of dimension needs to be reversed.

Memory Model

The CUDA and SYCL memory model is a hierarchical structure. In CUDA, it comprises multiple memory spaces, such as shared memory, global memory, constant memory, and unified memory. Shared memory allows for efficient communication within a thread block. Global memory is accessible to all threads across the device, providing a larger storage space but slower access compared to shared memory. Constant memory is a read-only space, suitable for storing unchanging data such as constants or lookup tables. Unified memory can be accessed both on host and device.

Similarly, in SYCL, local memory is shared within a work-group, global memory is accessible by all work-items, and shared memory is accessible on the host and device. According to the SYCL 2020 Specification, constant memory no longer appears in the SYCL device memory model in SYCL 2020.

Shared Memory

CUDA shared memory can be mapped to SYCL local memory. To perform this migration, declare an accessor with access target set to `sycl::access::target::local`. For example:

Original CUDA Code	Migrated SYCL Code

`__global__ void foo() {`
`  __shared__ int shm[16];`
`  shm[0] = 2;`
`}`
 	
`int main() {`
`  foo<<<1, 1>>>();`
`}`
	`void foo(int *shm) {`
	` shm[0] = 2;`
	`}`
	
	`int main() {`
	`  sycl::queue q;`
	`  q.submit([&](sycl::handler &cgh) {`

Original CUDA Code	Migrated SYCL Code

	`sycl::local_accessor<int> shm_acc(sycl::range<1>(16), cgh);`
	`cgh.parallel_for(`
	`sycl::nd_range<3>(sycl::range<3>(1, 1, 1), sycl::range<3>(1, 1, 1)), [=] (sycl::nd_item<3> item_ct1) {`
	`foo(shm_acc.get_pointer());`
	`});`
	`});`
	`}`

Global, Constant, and Unified Memory

CUDA global memory and constant memory can be mapped to SYCL global memory. CUDA unified memory can be mapped to SYCL shared memory. To perform this migration, allocate memory through `sycl::malloc_device` or `sycl::malloc_shared`. For example:

Original CUDA Code	Migrated SYCL Code

`void foo() {`
`int *mem1, *mem2;`
 	
`cudaMalloc(&mem1, 10);`
`cudaMallocManaged(&mem2, 10);`
`}`
	`void foo() {`
	`sycl::queue q;`
	`int *mem1, *mem2;`
	
	`mem1 = sycl::malloc_device<int>(10, q);`
	`mem2 = sycl::malloc_shared<int>(10, q);`
	`}`

Note that in CUDA, type specifiers `__device__`, `__constant__`, and `__managed__` can be used to declare a variable resident in global memory and unified memory. There is no direct equivalence in SYCL, but you can implement similar functionality with helper classes. A reference implementation for `dpct::global_memory`, `dpct::constant_memory`, and `dpct::shared_memory` is provided in the SYCLomatic project.

CUDA Device API Mapping

Synchronization API

In CUDA, synchronization functions are used to coordinate the execution of different threads in a CUDA kernel. `__syncthreads()` blocks the execution of all threads in a thread block until all threads in that block have reached the function. Additionally, all global and shared memory accesses made by these threads prior to `__syncthreads()` are visible to all threads in the block. The CUDA `__syncthreads` function can be mapped to `sycl::group_barrier` (with a `sycl::group` object passed in). For the CUDA `__syncthreads_and`, `__syncthreads_or`, and `__syncthreads_count` function migration, an additional group algorithm is needed after `sycl::group_barrier`. The CUDA `__syncwarp` function can be mapped to `sycl::group_barrier` (with a `sycl::sub_group` object spassed in).

The following table shows CUDA to SYCL mapping for synchronization functions:

CUDA	SYCL
`__syncthreads()`	`sycl::group_barrier(Group)`
`__syncthreads_and()`	`sycl::group_barrier(Group)` and `sycl::all_of_group(Group, predicate)`

CUDA	SYCL
`__syncthreads_or()`	`sycl::group_barrier(Group)` and `sycl::any_of_group(Group, predicate)`
`__syncthreads_count()`	`sycl::group_barrier(Group)` and `sycl::reduce_over_group(Group, predicate? 1:0, sycl::ext::oneapi::plus<>())`
`__syncwarp()`	`sycl::group_barrier(Sub_group)`

Memory Fence API

Memory fence functions can be used to enforce some ordering on memory accesses. The memory fence functions differ in the scope in which the orderings are enforced. CUDA memory fence functions can be mapped to `sycl::atomic_fence` with different memory scope.

The following table shows CUDA to SYCL mapping for fence functions:

CUDA	SYCL
`__threadfence_block()`	`sycl::atomic_fence(sycl::memory_order::acq_rel, sycl::memory_scope::work_group)`
`__threadfence()`	`sycl::atomic_fence(sycl::memory_order::acq_rel, sycl::memory_scope::device)`
`__threadfence_system()`	`sycl::atomic_fence(sycl::memory_order::acq_rel, sycl::memory_scope::system)`

Warp Intrinsic API

CUDA warp intrinsic functions, including warp vote and shuffle functions, can be mapped to the SYCL group algorithm API.

Warp Vote API

The CUDA warp vote API can map to the SYCL group algorithm API, as shown in the following table:

CUDA	SYCL
`__all()/__all_sync()`	`sycl::all_of_group()`
`__any()/__any_sync()`	`sycl::any_of_group()`
`__ballot()/__ballot_sync()`	`sycl::reduce_over_group()`

For the sync version of CUDA warp intrinsic functions, a mask is passed that specifies the threads participating in the call. The equivalent SYCL API does not support mask directly. Refer to the reference implementation of mask version APIs in the SYCLomatic project.

The following table shows the original CUDA code for a warp vote example, migrated to SYCL:

Original CUDA Code	Migrated SYCL Code

`__device__ void foo(){`
`  __all_sync(mask, predicate);`
`}`
	`void foo(sycl::nd_item<3> item) {`
	`  auto g = item.get_sub_group();`
	`  sycl::all_of_group(g, (~mask & (0x1 <<`
	`g.get_local_linear_id())) || predicate);`
	`}`

Warp Shuffle API

CUDA warp shuffle functions can be mapped to the following SYCL group algorithms:

CUDA	SYCL
`__shfl()/__shfl_sync()`	`sycl::select_from_group()`
`__shfl_up()/__shfl_up_sync()`	`sycl::shift_group_right()`
`__shfl_down()/__shfl_down_sync()`	`sycl::shift_group_left()`
`__shfl_xor()/__shfl_xor_sync()`	`sycl::permute_group_by_xor()`

CUDA shuffle functions support operate on a subset threads of warp. The equivalent SYCL API does not support operations on a subset of sub_group directly. Refer to the helper implementation in the SYCLomatic project.

The following table shows the original CUDA code for a warp shuffle example, migrated to SYCL:

Original CUDA Code	Migrated SYCL Code

`__device__ void foo(){`
`  __shfl_sync(mask, val, r_id, 16);`
`}`
	`void foo(sycl::nd_item<3> item) {`
	`  auto g = item.get_sub_group();`
	`  unsigned int start_index =`
	`(g.get_local_linear_id() / 16) * 16;`
	`  sycl::select_from_group(g, val, start_index`
	`+ r_id % 16);`
	`}`

CUDA Host API Mapping

Device Management

The CUDA device management API can map to the SYCL device class and its member functions as follows:

CUDA	SYCL
`cudaGetDeviceCount()`	`sycl::device::get_devices()`
`cudaSetDevice()`	`sycl::device dev { device_selector }` // Select sycl::device and make it ready for creating sycl::queue
`cudaGetDevice()`	`sycl::queue.get_device()` //Get active device from sycl::queue created
`cudaGetDeviceProperties()/` `cudaDeviceGetAttribute()`	`sycl::device.get_info<info type>()`

The following table shows the original CUDA code for a device management example, migrated to SYCL:

Original CUDA Code	Migrated SYCL Code

`int device_count;`
`cudaGetDeviceCount(&device_count);`
`for(int i = 0; i < device_count; i++) {`
`  cudaSetDevice(i);`
`  cudaDeviceProp prop;`
`  cudaGetDeviceProperties(&prop, i);`
`  int warp_size = prop.warpSize;`
`  …`
`  kernel<<<size_1, size_2>>>();`
`}`
	`auto devices =`
	`sycl::device::get_devices(sycl::info::device_t`
	`ype::gpu);`
	`for(auto &device : devices) {`
	`  sycl::queue q(device);`
	`  auto sub_group_sizes =`
	`device.get_info<sycl::info::device::sub_group_`
	`sizes>();`
	`  ...`
	`  q.parallel_for(sycl::nd_range<3>(size_1 *`
	`size_2, size_2), [=](sycl::nd_item<3> item){`
	`  kernel(item);`
	`  });`
	`}`

Stream Management

The CUDA stream management API can map to the SYCL queue class and its member functions as follows:

CUDA	SYCL
`cudaStreamCreate()`	Constructor of `sycl::queue`
`cudaStreamDestroy()`	Destructor of `sycl::queue`
`cudaStreamAddCallback()`	`std::async()`
`cudaStreamSynchronize()`	`sycl::queue.wait()`
`cudaStreamWaitEvent()`	`sycl::queue.ext_oneapi_submit_barrier()`

The following table shows the original CUDA code for a stream management example, migrated to SYCL:

Original CUDA Code	Migrated SYCL Code

`void callback(cudaStream_t st, cudaError_t`
`status, void *vp) {…}`
 	
`void test() {`
`  cudaStream_t stream;`
`  cudaEvent_t event;`
 	
`  cudaStreamCreate(&stream);`
`  cudaStreamAddCallback(stream, callback, 0,`
`0);`
`  cudaStreamSynchronize(stream);`
`  cudaStreamWaitEvent(stream, event, 0);`
`  cudaStreamDestroy(stream);`
`}`
	`void callback(sycl::queue st, int status,`
	`void *vp) {…}`
	
	`void test() {`
	`  sycl::queue q;`
	`  sycl::event event;`
	
	` std::async([&]() {`
	`  q.wait();`
	`  callback(q, 0, 0);`
	`  });`
	`  q.wait();`
	`  q.ext_oneapi_submit_barrier({event});`
	`}`

Note that the constructor of `sycl::queue` creates a queue with an out-of-order property by default. Use `sycl::property::queue::in_order::in_order()` in the construction of the queue to create an in-order queue.

Memory Management

The CUDA memory management API can map to the SYCL USM pointer-based memory management API as follows:

CUDA	SYCL
`cudaMalloc()`	`sycl::malloc_device()`
`cudaMallocHost()`	`sycl::malloc_host()`
`cudaMallocManaged()`	`sycl::malloc_shared()`
`cudaMemcpy()`	`sycl::queue.memcpy()`
`cudaMemset()`	`sycl::queue.memset()`
`cudaFree()/cudaFreeHost()`	`sycl::free()`

The following table shows the original CUDA code for a memory management example, migrated to SYCL:

Original CUDA Code

```
void test() {
  int *dev_ptr, *host_ptr, *shared_ptr;
  int size;

  ...

  cudaMalloc(&dev_ptr, size);
  cudaMallocHost(&host_ptr, size);
  cudaMallocManaged(&shared_ptr, size);
  cudaMemset(dev_ptr, size, 0);
  cudaMemcpy(host_ptr, dev_ptr, size,
cudaMemcpyHostToDevice);
  cudaMemcpy(shared_ptr, host_ptr, size,
cudaMemcpyHostToDevice);
  ...
  int a = shared_ptr[0];
  ...
  cudaFree(dev_ptr);
  cudaFree(host_ptr);
  cudaFree(shared_ptr);
}
```

Migrated SYCL Code

```
void test() {
  sycl::queue q;
  int *dev_ptr, *host_ptr, *shared_ptr;
  int size;
  ...
  dev_ptr = (int *)sycl::malloc_device(size,
q);
  host_ptr = (int *)sycl::malloc_host(size,
q);
  shared_ptr = (int
*)sycl::malloc_shared(size, q);
  q.memset(dev_ptr, size, 0).wait();
  q.memcpy(host_ptr, dev_ptr, size).wait();
  q.memcpy(shared_ptr, host_ptr, size).wait();
  ...
  int a = shared_ptr[0];
  ...
  sycl::free(dev_ptr, q);
  sycl::free(host_ptr, q);
  sycl::free(shared_ptr, q);
}
```

Error Handling

In the CUDA runtime library error handling relies mainly on the error code returned by the API call. In SYCL, synchronous errors are reported immediately by the runtime throwing an exception. Use try-catch statements to catch and process these exceptions. For example:

Original CUDA Code	Migrated SYCL Code

`void test() {`
`  int *ptr;`
`  if (cudaMalloc(&ptr, sizeof(int))) {`
`  std::cout << "error" << std::endl;`
`  }`
`}`
	`void test() try {`
	`  int *ptr;`
	`  sycl::queue q;`
	`  ptr = sycl::malloc_device<int>(1, q);`
	`}`
	`catch (sycl::exception const &exc) {`
	`  std::cerr << exc.what() << "Exception caught at file:" << __FILE__ << ", line:" << __LINE__ << std::endl;`
	`  std::exit(1);`
	`}`

In SYCL, asynchronous errors are not reported immediately as they occur. The queue can optionally take an asynchronous handler at construction, with an `exception_list` as a parameter. Invocation of an `async_handler` may be triggered by the queue member functions `queue::wait_and_throw()`, `queue::throw_asynchronous()`, or automatically on destruction of a queue that contains unconsumed asynchronous errors. When invoked, an `async_handler` is called and receives an `exception_list` argument containing a list of exception objects representing any unconsumed asynchronous errors associated with the queue or context. The following example shows one implementation of an asynchronous exception handler, migrated to SYCL:

Original CUDA Code	Migrated SYCL Code

`void test() {`
`  int *ptr;`
`  kernel<<<1, 1>>>();`
`  if (cudaDeviceSynchronize()) {`
`  std::cout << "error" << std::endl;`
`  }`
`}`
	`auto exception_handler = []`
	`(cl::sycl::exception_list exceptions) {`
	`  for (std::exception_ptr const &e : exceptions) {`
	`  try {`
	`  std::rethrow_exception(e);`
	`  } catch (cl::sycl::exception const &e) {`
	`  std::cerr << "Caught asynchronous SYCL exception:" << std::endl`
	`  << e.what() << std::endl`
	`  << "Exception caught at file:" << __FILE__`
	`  << ", line:" << __LINE__ << std::endl;`
	`  }`
	`  }`
	`};`
	`void test() {`
	`  sycl::queue q{exception_handler};`
	`  q.parallel_for(`
	`  sycl::nd_range<3>(size_1 * size_2, size_2),`
	`  [=](sycl::nd_item<3> item){`
	`  kernel(item);`
	`  }).wait_and_throw();`
	`}`

CUDA* to SYCL* Term Mapping Quick Reference

This quick reference maps common CUDA* terms to SYCL* terms, to help start the migration process.

Architecture Terminology Mapping

CUDA Capable GPU	SYCL Capable GPU from Intel	
	Xe-LP and prior generations	**Xe-HPG and Xe-HPC**
CUDA Core	Execution Unit (EU)	Vector Engine & Matrix Engine (XVE & XMX)
Streaming Multiprocessor (SM)	Subslice(SS) or Dual Subslice(DSS)	Xe-Core
Processor Clusters (PC)	Slice	Xe-Slice
N/A	N/A	Xe-Stack

Execution Model Mapping

CUDA	SYCL
Thread	Work-item
Warp	Sub-group
Block	Work-group
Grid	ND-range

Memory Model Mapping

CUDA	SYCL
Register	Private memory
Shared memory	Local memory
Constant memory	N/A
Global memory	Global memory

Memory Specifier Mapping

CUDA	SYCL
`__device__`	N/A
`__shared__`	N/A
`__constant__`	N/A
`__managed__`	N/A

Function Execution Space Specifiers Mapping

CUDA	SYCL
`__global__`	Not required

CUDA	SYCL
`__device__`	Not required
`__host__`	Not required

Mapping of Key Host Type Used to Offload Task

CUDA	SYCL
device	`sycl::device`
stream	`sycl::queue`
event	`sycl::event`

Kernel Execution Configures Mapping

CUDA	SYCL
`dim3`	`sycl::range<3>`

`Kernel<<<gridDim, blockDim>>>(...)`	**1.** Member function `parallel_for` of class `sycl::queue`:
	`sycl::queue q;`
	`q.parallel_for(sycl::nd_range<3>(girdDim * blockDim, blockDim), [=](sycl::nd_item<3> item){`
	`  kernel(...);`
	`});`
	2. Member function `submit` of class `sycl::queue`:
	`sycl::queue q;`
	`q.submit([&](sycl::handler &cgh){`
	`  cgh.parallel_for(sycl::nd_range<3>(gridDim * blockDim, blockDim), [=](sycl::nd_item<3> item){`
	`  kernel(...);`
	`  });`
	`});`

Built-In Execution Space Index Mapping

CUDA	SYCL
`gridDim.x/y/z`	`sycl::nd_item.get_group_range(2/1/0)`

CUDA	SYCL
`blockDim.x/y/z`	`sycl::nd_item.get_local _range().get(2/1/0)`
`blockIdx.x/y/z`	`sycl::nd_item.get_group(2/1/0)`
`threadIdx.x/y/z`	`sycl::nd_item.get_local_id(2/1/0)`
`warpSize`	`sycl::nd_item. .get_sub_group().get_local _range().get(0)`

CUDA API Migration Support

Migration support status updated on 10/31/2024.

The following tables provide the current status for CUDA* API migration support.

ASM API

Function	Migration Support	Diagnostic Message
abs	YES	
activemask	NO	
add	YES	Partial
addc	NO	
alloca	NO	
and	YES	
applypriority	NO	
atom	YES	Partial
bar	YES	Partial
barrier	NO	
bfe	YES	
bfi	YES	
bfind	NO	
bmsk	NO	
bra	NO	
brev	YES	
brkpt	NO	
brx	NO	

Function	Migration Support	Diagnostic Message
call	NO	
clz	YES	
cnot	YES	
copysign	YES	
cos	YES	
cp	NO	
createpolicy	NO	
cvt	YES	Partial
cvta	NO	
discard	NO	
div	YES	Partial
dp2a	YES	
dp4a	YES	
elect	NO	
ex2	YES	
exit	NO	
fence	NO	
fma	YES	Partial
fns	NO	
getctarank	NO	
griddepcontrol	NO	
isspacep	NO	
istypep	NO	
ld	YES	Partial
ldmatrix	NO	
ldu	NO	
lg2	YES	
lop3	YES	
mad	YES	Partial
mad24	YES	Partial

Function	Migration Support	Diagnostic Message
madc	NO	
mapa	NO	
match	NO	
max	YES	
mbarrier	NO	
membar	NO	
min	YES	
mma	NO	
mov	YES	
movmatrix	NO	
mul	YES	Partial
mul24	YES	Partial
multimem	NO	
nanosleep	NO	
neg	YES	
not	YES	
or	YES	
pmevent	NO	
popc	YES	
prefetch	NO	
prefetchu	NO	
prmt	NO	
rcp	YES	
red	NO	
redux	NO	
rem	YES	
ret	YES	
rsqrt	YES	
sad	YES	
selp	YES	

Function	Migration Support	Diagnostic Message
set	NO	
setmaxnreg	NO	
setp	YES	
shf	NO	
shfl	NO	
shl	YES	
shr	YES	
sin	YES	
slct	YES	Partial
sqrt	YES	
st	YES	Partial
stackrestore	NO	
stacksave	NO	
stmatrix	NO	
sub	YES	Partial
subc	NO	
suld	NO	
suq	NO	
sured	NO	
sust	NO	
szext	NO	
tanh	YES	
testp	YES	
tex	NO	
tld4	NO	
trap	NO	
txq	NO	
vabsdiff	YES	Partial
vabsdiff2	YES	Partial
vabsdiff4	YES	Partial

Function	Migration Support	Diagnostic Message
vadd	YES	Partial
vadd2	YES	Partial
vadd4	YES	Partial
vavrg2	YES	Partial
vavrg4	YES	Partial
vmad	NO	
vmax	YES	Partial
vmax2	YES	Partial
vmax4	YES	Partial
vmin	YES	Partial
vmin2	YES	Partial
vmin4	YES	Partial
vote	NO	
vset	YES	Partial
vset2	YES	Partial
vset4	YES	Partial
vshl	YES	Partial
vshr	YES	Partial
vsub	YES	Partial
vsub2	YES	Partial
vsub4	YES	Partial
wgmma	NO	
wmma	NO	
xor	YES	Partial

CUB API

Function	Migration Support	Diagnostic Message
cub::ShuffleUp	YES	
cub::ShuffleDown	YES	
cub::ShuffleIndex	YES	

Function	Migration Support	Diagnostic Message
cub::WarpScan::InclusiveSum	YES	
cub::WarpScan::ExclusiveSum	YES	
cub::WarpScan::InclusiveScan	YES	
cub::WarpScan::ExclusiveScan	YES	
cub::WarpScan::Scan	NO	
cub::WarpScan::Broadcast	YES	
cub::WarpReduce::Sum	YES	
cub::WarpReduce::HeadSegmentedSum	NO	
cub::WarpReduce::TailSegmentedSum	NO	
cub::WarpReduce::Reduce	YES	
cub::WarpReduce::HeadSegmentedReduce	NO	
cub::WarpReduce::TailSegmentedReduce	NO	
cub::WarpExchange::BlockedToStriped	NO	
cub::WarpExchange::StripedToBlocked	NO	
cub::WarpExchange::ScatterToStriped	NO	
cub::WarpLoad::Load	NO	
cub::WarpStore::Store	NO	
cub::BlockAdjacentDifference::SubtractLeft	NO	
cub::BlockAdjacentDifference::SubtractLeftPartialTile	NO	
cub::BlockAdjacentDifference::SubtractRight	NO	
cub::BlockAdjacentDifference::SubtractRightPartialTile	NO	
cub::BlockAdjacentDifference::FlagHeads	NO	
cub::BlockAdjacentDifference::FlagTails	NO	
cub::BlockAdjacentDifference::FlagHeadsAndTails	NO	
cub::BlockDiscontinuity::FlagHeads	NO	
cub::BlockDiscontinuity::FlagTails	NO	

Function	Migration Support	Diagnostic Message
cub::BlockDiscontinuity::FlagHeadsAndTails	NO	
cub::BlockExchange::StripedToBlocked	YES	
cub::BlockExchange::BlockedToStriped	YES	
cub::BlockExchange::WarpStripedToBlocked	NO	
cub::BlockExchange::BlockedToWarpStriped	NO	
cub::BlockExchange::ScatterToBlocked	YES	
cub::BlockExchange::ScatterToStriped	YES	
cub::BlockExchange::ScatterToStripedGuarded	NO	
cub::BlockExchange::ScatterToStripedFlagged	NO	
cub::BlockHistogram::InitHistogram	NO	
cub::BlockHistogram::Histogram	NO	
cub::BlockHistogram::Composite	NO	
cub::BlockLoad::Load	YES	Partial
cub::BlockStore::Store	YES	Partial
cub::BlockRadixSort::Sort	YES	Partial
cub::BlockRadixSort::SortDescending	YES	Partial
cub::BlockRadixSort::SortBlockedToStriped	YES	Partial
cub::BlockRadixSort::SortDescendingBlockedToStriped	YES	Partial
cub::BlockReduce::Reduce	YES	
cub::BlockReduce::Sum	YES	
cub::BlockScan::ExclusiveSum	YES	
cub::BlockScan::ExclusiveScan	YES	
cub::BlockScan::InclusiveSum	YES	
cub::BlockScan::InclusiveScan	YES	
cub::BlockShuffle::Offset	NO	
cub::BlockShuffle::Rotate	NO	

Function	Migration Support	Diagnostic Message
cub::BlockShuffle::Up	NO	
cub::BlockShuffle::Down	NO	
cub::BlockDiscontinuity::FlagHeads	NO	
cub::BlockDiscontinuity::FlagTails	NO	
cub::BlockDiscontinuity::FlagHeadsAndTails	NO	
cub::BlockMergeSortStrategy::get_linear_tid	NO	
cub::BlockMergeSortStrategy::Sort	NO	
cub::BlockMergeSortStrategy::StableSort	NO	
cub::BlockRadixRank::RankKeys	NO	
cub::BlockRadixRankMatch::RankKeys	NO	
cub::BlockRadixRankMatch::CallBack	NO	
cub::RadixSortTwiddle::In	NO	
cub::RadixSortTwiddle::Out	NO	
cub::RadixSortTwiddle::DefaultKey	NO	
cub::ShiftDigitExtractor::Digit	NO	
cub::BFEDigitExtractor::Digit	NO	
cub::BFEDigitExtractor::ProcessFloatMinusZero	NO	
cub::BaseDigitExtractor::ProcessFloatMinusZero	NO	
cub::DeviceAdjacentDifference::SubtractLeftCopy	NO	
cub::DeviceAdjacentDifference::SubtractLeft	NO	
cub::DeviceAdjacentDifference::SubtractRightCopy	NO	
cub::DeviceAdjacentDifference::SubtractRight	NO	
cub::DeviceHistogram::HistogramEven	YES	DPCT1026 / DPCT1027
cub::DeviceHistogram::MultiHistogramEven	YES	DPCT1026 / DPCT1027

Function	Migration Support	Diagnostic Message
cub::DeviceHistogram::HistogramRange	YES	DPCT1026 / DPCT1027
cub::DeviceHistogram::MultiHistogramRange	YES	DPCT1026 / DPCT1027
cub::DevicePartition::Flagged	YES	DPCT1026 / DPCT1027
cub::DevicePartition::If	YES	DPCT1026 / DPCT1027
cub::DeviceMergeSort::SortPairs	YES	DPCT1026 / DPCT1027
cub::DeviceMergeSort::SortPairsCopy	NO	
cub::DeviceMergeSort::SortKeys	YES	DPCT1026 / DPCT1027
cub::DeviceMergeSort::SortKeysCopy	YES	DPCT1026 / DPCT1027
cub::DeviceMergeSort::StableSortPairs	YES	DPCT1026 / DPCT1027
cub::DeviceMergeSort::StableSortKeys	YES	DPCT1026 / DPCT1027
cub::DeviceRadixSort::SortPairs	YES	DPCT1026 / DPCT1027
cub::DeviceRadixSort::SortPairsDescending	YES	DPCT1026 / DPCT1027
cub::DeviceRadixSort::SortKeys	YES	DPCT1026 / DPCT1027
cub::DeviceRadixSort::SortKeysDescending	YES	DPCT1026 / DPCT1027
cub::DeviceReduce::Reduce	YES	DPCT1026 / DPCT1027
cub::DeviceReduce::Sum	YES	DPCT1026 / DPCT1027
cub::DeviceReduce::Min	YES	DPCT1026 / DPCT1027
cub::DeviceReduce::ArgMin	YES	DPCT1026 / DPCT1027
cub::DeviceReduce::Max	YES	DPCT1026 / DPCT1027
cub::DeviceReduce::ArgMax	YES	DPCT1026 / DPCT1027
cub::DeviceReduce::ReduceByKey	YES	DPCT1026 / DPCT1027
cub::DeviceRunLengthEncode::Encode	YES	DPCT1026 / DPCT1027
cub::DeviceRunLengthEncode::NonTrivialRuns	YES	DPCT1026 / DPCT1027
cub::DeviceScan::ExclusiveSum	YES	DPCT1026 / DPCT1027
cub::DeviceScan::ExclusiveScan	YES	DPCT1026 / DPCT1027
cub::DeviceScan::InclusiveSum	YES	DPCT1026 / DPCT1027
cub::DeviceScan::InclusiveScan	YES	DPCT1026 / DPCT1027
cub::DeviceScan::InclusiveSumByKey	YES	DPCT1026 / DPCT1027

Function	Migration Support	Diagnostic Message
cub::DeviceScan::ExclusiveSumByKey	YES	DPCT1026 / DPCT1027
cub::DeviceScan::InclusiveScanByKey	YES	DPCT1026 / DPCT1027
cub::DeviceScan::ExclusiveScanByKey	YES	DPCT1026 / DPCT1027
cub::DeviceSelect::Flagged	YES	DPCT1026 / DPCT1027
cub::DeviceSelect::If	YES	DPCT1026 / DPCT1027
cub::DeviceSelect::Unique	YES	DPCT1026 / DPCT1027
cub::DeviceSelect::UniqueByKey	YES	DPCT1026 / DPCT1027
cub::DeviceSpmv::CsrMV	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedRadixSort::SortPairs	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedRadixSort::SortPairsDescending	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedRadixSort::SortKeys	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedRadixSort::SortKeysDescending	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedReduce::Reduce	YES	DPCT1026 / DPCT1027 / DPCT1091 / DPCT1092
cub::DeviceSegmentedReduce::Sum	YES	DPCT1026 / DPCT1027 / DPCT1092
cub::DeviceSegmentedReduce::Min	YES	DPCT1026 / DPCT1027 / DPCT1092
cub::DeviceSegmentedReduce::ArgMin	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedReduce::Max	YES	DPCT1026 / DPCT1027 / DPCT1092
cub::DeviceSegmentedReduce::ArgMax	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedSort::SortKeys	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedSort::SortKeysDescending	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedSort::StableSortKeys	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedSort::StableSortKeysDescending	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedSort::SortPairs	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedSort::SortPairsDescending	YES	DPCT1026 / DPCT1027

Function	Migration Support	Diagnostic Message
cub::DeviceSegmentedSort::StableSortPairs	YES	DPCT1026 / DPCT1027
cub::DeviceSegmentedSort::StableSortPairsDescending	YES	DPCT1026 / DPCT1027
cub::DeviceCopy::Batched	NO	
cub::DeviceFor::Bulk	NO	
cub::DeviceFor::ForEachN	NO	
cub::DeviceFor::ForEach	NO	
cub::DeviceFor::ForEachCopy	NO	
cub::DeviceFor::ForEachCopyN	NO	
cub::DeviceMemcpy::Batched	NO	
cub::GridBarrier::Sync	NO	
cub::GridBarrierLifetime::Sync	NO	
cub::GridBarrierLifetime::HostReset	NO	
cub::GridBarrierLifetime::Setup	NO	
cub::GridQueue::FillAndResetDrain	NO	
cub::GridQueue::ResetDrain	NO	
cub::GridQueue::ResetFill	NO	
cub::GridQueue::FillSize	NO	
cub::GridQueue::Drain	NO	
cub::GridQueue::Fill	NO	
cub::GridQueue::AllocationSize	NO	
cub::GridEvenShare::DispatchInit	NO	
cub::GridEvenShare::BlockInit	NO	
cub::ThreadLoad	YES	
cub::ThreadStore	YES	
cub::LoadDirectBlocked	YES	
cub::LoadDirectBlockedVectorized	NO	
cub::StoreDirectBlocked	YES	
cub::StoreDirectBlockedVectorized	NO	
cub::LoadDirectStriped	YES	

Function	Migration Support	Diagnostic Message
cub::StoreDirectStriped	YES	
cub::LoadDirectWarpStriped	NO	
cub::StoreDirectWarpStriped	NO	
cub::SHR_ADD	YES	
cub::SHL_ADD	YES	
cub::BFE	YES	
cub::BFI	YES	
cub::IADD3	YES	
cub::PRMT	NO	
cub::ThreadExit	NO	
cub::ThreadTrap	NO	
cub::RowMajorTid	YES	
cub::LaneId	YES	
cub::WarpId	YES	
cub::LaneMaskLt	NO	
cub::LaneMaskLe	NO	
cub::LaneMaskGt	NO	
cub::LaneMaskGe	NO	
cub::Swap	NO	
cub::UpperBound	NO	
cub::WarpMask	NO	
cub::StableOddEvenSort	NO	
cub::SerialMerge	NO	
cub::Nominal8BItemsToItems	NO	
cub::Nominal4BItemsToItemsCombined	NO	
cub::Nominal4BItemsToItems	NO	
cub::MidPoint	NO	
cub::MergePathSearch	NO	
cub::MergePath	NO	
cub::MatchAny	NO	

Function	Migration Support	Diagnostic Message
cub::MakeBinaryFlip	NO	
cub::LowerBound	NO	
cub::GetPerDeviceAttributeCache	NO	
cub::DivideAndRoundUp	NO	
cub::KernelConfig::Init	NO	
cub::CachingDeviceAllocator::SetMaxCachedBytes	NO	
cub::CachingDeviceAllocator::DeviceAllocate	NO	
cub::CachingDeviceAllocator::DeviceFree	NO	
cub::CachingDeviceAllocator::FreeAllCached	NO	
cub::ChainedPolicy::Invoke	NO	
cub::Debug	NO	
cub::MaxSmOccupancy	NO	
cub::PtxVersion	YES	
cub::SmVersion	YES	
cub::CurrentDevice	YES	
cub::DeviceCount	YES	
cub::DeviceCountUncached	YES	
cub::DeviceCountCachedValue	YES	
cub::PtxVersionUncached	YES	
cub::SmVersionUncached	YES	
cub::SyncStream	YES	
cub::ArgIndexInputIterator::normalize	YES	
cub::TexObjInputIterator::BindTexture	NO	
cub::TexObjInputIterator::UnbindTexture	NO	
cub::TexRefInputIterator::BindTexture	NO	
cub::TexRefInputIterator::UnbindTexture	NO	
cub::CountingInputIterator	YES	

Function	Migration Support	Diagnostic Message
cub::TransformInputIterator	YES	
cub::ConstantInputIterator	YES	
cub::DiscardOutputIterator	YES	
cub::CacheModifiedInputIterator	NO	
cub::CacheModifiedOutputIterator	NO	

cuBLAS API

Function	Migration Support	Diagnostic Message
cublasCreate	YES	
cublasDestroy	YES	
cublasGetVersion	YES	
cublasGetProperty	NO	
cublasSetStream	YES	
cublasGetStream	YES	
cublasGetPointerMode	YES	DPCT1026 / DPCT1027
cublasSetPointerMode	YES	DPCT1026 / DPCT1027
cublasSetVector	YES	DPCT1018 / DPCT1020
cublasGetVector	YES	DPCT1018 / DPCT1020
cublasSetMatrix	YES	DPCT1018 / DPCT1020
cublasGetMatrix	YES	DPCT1018 / DPCT1020
cublasSetVectorAsync	YES	DPCT1018 / DPCT1020
cublasGetVectorAsync	YES	DPCT1018 / DPCT1020
cublasSetMatrixAsync	YES	DPCT1018 / DPCT1020
cublasGetMatrixAsync	YES	DPCT1018 / DPCT1020
cublasSetAtomicsMode	YES	DPCT1026 / DPCT1027
cublasGetAtomicsMode	YES	DPCT1026 / DPCT1027
cublasSetMathMode	YES	
cublasGetMathMode	YES	
cublasLoggerConfigure	NO	
cublasGetLoggerCallback	NO	

Function	Migration Support	Diagnostic Message
cublasSetLoggerCallback	NO	
cublasIsamax	YES	DPCT1020 / DPCT1034
cublasIdamax	YES	DPCT1020 / DPCT1034
cublasIcamax	YES	DPCT1020 / DPCT1034
cublasIzamax	YES	DPCT1020 / DPCT1034
cublasIsamin	YES	DPCT1020 / DPCT1034
cublasIdamin	YES	DPCT1020 / DPCT1034
cublasIcamin	YES	DPCT1020 / DPCT1034
cublasIzamin	YES	DPCT1020 / DPCT1034
cublasSasum	YES	DPCT1020 / DPCT1034
cublasDasum	YES	DPCT1020 / DPCT1034
cublasScasum	YES	DPCT1020 / DPCT1034
cublasDzasum	YES	DPCT1020 / DPCT1034
cublasSaxpy	YES	DPCT1020
cublasDaxpy	YES	DPCT1020
cublasCaxpy	YES	DPCT1020
cublasZaxpy	YES	DPCT1020
cublasScopy	YES	DPCT1020
cublasDcopy	YES	DPCT1020
cublasCcopy	YES	DPCT1020
cublasZcopy	YES	DPCT1020
cublasSdot	YES	DPCT1020 / DPCT1034
cublasDdot	YES	DPCT1020 / DPCT1034
cublasCdotu	YES	DPCT1020 / DPCT1034
cublasCdotc	YES	DPCT1020 / DPCT1034
cublasZdotu	YES	DPCT1020 / DPCT1034
cublasZdotc	YES	DPCT1020 / DPCT1034
cublasSnrm2	YES	DPCT1020 / DPCT1034
cublasDnrm2	YES	DPCT1020 / DPCT1034
cublasScnrm2	YES	DPCT1020 / DPCT1034

Function	Migration Support	Diagnostic Message
cublasDznrm2	YES	DPCT1020 / DPCT1034
cublasSrot	YES	DPCT1020
cublasDrot	YES	DPCT1020
cublasCrot	YES	DPCT1020
cublasCsrot	YES	DPCT1020
cublasZrot	YES	DPCT1020
cublasZdrot	YES	DPCT1020
cublasSrotg	YES	DPCT1020 / DPCT1034
cublasDrotg	YES	DPCT1020 / DPCT1034
cublasCrotg	YES	DPCT1020 / DPCT1034
cublasZrotg	YES	DPCT1020 / DPCT1034
cublasSrotm	YES	DPCT1020 / DPCT1034
cublasDrotm	YES	DPCT1020 / DPCT1034
cublasSrotmg	YES	DPCT1020 / DPCT1034
cublasDrotmg	YES	DPCT1020 / DPCT1034
cublasSscal	YES	DPCT1020
cublasDscal	YES	DPCT1020
cublasCscal	YES	DPCT1020
cublasCsscal	YES	DPCT1020
cublasZscal	YES	DPCT1020
cublasZdscal	YES	DPCT1020
cublasSswap	YES	DPCT1020
cublasDswap	YES	DPCT1020
cublasCswap	YES	DPCT1020
cublasZswap	YES	DPCT1020
cublasSgbmv	YES	DPCT1020
cublasDgbmv	YES	DPCT1020
cublasCgbmv	YES	DPCT1020
cublasZgbmv	YES	DPCT1020
cublasSgemv	YES	DPCT1020

Function	Migration Support	Diagnostic Message
cublasDgemv	YES	DPCT1020
cublasCgemv	YES	DPCT1020
cublasZgemv	YES	DPCT1020
cublasSger	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDger	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCgeru	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCgerc	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZgeru	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZgerc	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSsbmv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDsbmv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSspmv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDspmv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSspr	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDspr	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSspr2	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDspr2	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSsymv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDsymv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCsymv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZsymv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSsyr	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDsyr	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCsyr	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZsyr	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSsyr2	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDsyr2	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCsyr2	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZsyr2	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasStbmv	YES	DPCT1020

Function	Migration Support	Diagnostic Message
cublasDtbmv	YES	DPCT1020
cublasCtbmv	YES	DPCT1020
cublasZtbmv	YES	DPCT1020
cublasStbsv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDtbsv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCtbsv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZtbsv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasStpmv	YES	DPCT1020
cublasDtpmv	YES	DPCT1020
cublasCtpmv	YES	DPCT1020
cublasZtpmv	YES	DPCT1020
cublasStpsv	YES	DPCT1020
cublasDtpsv	YES	DPCT1020
cublasCtpsv	YES	DPCT1020
cublasZtpsv	YES	DPCT1020
cublasStrmv	YES	DPCT1020
cublasDtrmv	YES	DPCT1020
cublasCtrmv	YES	DPCT1020
cublasZtrmv	YES	DPCT1020
cublasStrsv	YES	DPCT1020
cublasDtrsv	YES	DPCT1020
cublasCtrsv	YES	DPCT1020
cublasZtrsv	YES	DPCT1020
cublasChemv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZhemv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasChbmv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZhbmv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasChpmv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZhpmv	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCher	YES	DPCT1020 / DPCT1034 / DPCT1041

Function	Migration Support	Diagnostic Message
cublasZher	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCher2	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZher2	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasChpr	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZhpr	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasChpr2	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZhpr2	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSgemm	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDgemm	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCgemm	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZgemm	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasHgemm	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCgemm3m	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZgemm3m	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasHgemmBatched	YES	DPCT1020
cublasSgemmBatched	YES	DPCT1020
cublasDgemmBatched	YES	DPCT1020
cublasCgemmBatched	YES	DPCT1020
cublasZgemmBatched	YES	DPCT1020
cublasHgemmStridedBatched	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSgemmStridedBatched	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDgemmStridedBatched	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCgemmStridedBatched	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCgemm3mStridedBatched	NO	
cublasZgemmStridedBatched	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSsymm	YES	DPCT1020
cublasDsymm	YES	DPCT1020
cublasCsymm	YES	DPCT1020
cublasZsymm	YES	DPCT1020
cublasSsyrk	YES	DPCT1020

Function	Migration Support	Diagnostic Message
cublasDsyrk	YES	DPCT1020
cublasCsyrk	YES	DPCT1020
cublasZsyrk	YES	DPCT1020
cublasSsyr2k	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasDsyr2k	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCsyr2k	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZsyr2k	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasSsyrkx	YES	DPCT1020
cublasDsyrkx	YES	DPCT1020
cublasCsyrkx	YES	DPCT1020
cublasZsyrkx	YES	DPCT1020
cublasStrmm	YES	DPCT1020
cublasDtrmm	YES	DPCT1020
cublasCtrmm	YES	DPCT1020
cublasZtrmm	YES	DPCT1020
cublasStrsm	YES	DPCT1020
cublasDtrsm	YES	DPCT1020
cublasCtrsm	YES	DPCT1020
cublasZtrsm	YES	DPCT1020
cublasStrsmBatched	YES	DPCT1020
cublasDtrsmBatched	YES	DPCT1020
cublasCtrsmBatched	YES	DPCT1020
cublasZtrsmBatched	YES	DPCT1020
cublasChemm	YES	DPCT1020
cublasZhemm	YES	DPCT1020
cublasCherk	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZherk	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCher2k	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasZher2k	YES	DPCT1020 / DPCT1034 / DPCT1041
cublasCherkx	YES	DPCT1020

Function	Migration Support	Diagnostic Message
cublasZherkx	YES	DPCT1020
cublasSgeam	YES	DPCT1020
cublasDgeam	YES	DPCT1020
cublasCgeam	YES	DPCT1020
cublasZgeam	YES	DPCT1020
cublasSdgmm	YES	DPCT1020
cublasDdgmm	YES	DPCT1020
cublasCdgmm	YES	DPCT1020
cublasZdgmm	YES	DPCT1020
cublasSgetrfBatched	YES	DPCT1020 / DPCT1047
cublasDgetrfBatched	YES	DPCT1020 / DPCT1047
cublasCgetrfBatched	YES	DPCT1020 / DPCT1047
cublasZgetrfBatched	YES	DPCT1020 / DPCT1047
cublasSgetrsBatched	YES	DPCT1020
cublasDgetrsBatched	YES	DPCT1020
cublasCgetrsBatched	YES	DPCT1020
cublasZgetrsBatched	YES	DPCT1020
cublasSgetriBatched	YES	DPCT1020
cublasDgetriBatched	YES	DPCT1020
cublasCgetriBatched	YES	DPCT1020
cublasZgetriBatched	YES	DPCT1020
cublasSmatinvBatched	NO	
cublasDmatinvBatched	NO	
cublasCmatinvBatched	NO	
cublasZmatinvBatched	NO	
cublasSgeqrfBatched	YES	DPCT1020
cublasDgeqrfBatched	YES	DPCT1020
cublasCgeqrfBatched	YES	DPCT1020
cublasZgeqrfBatched	YES	DPCT1020
cublasSgelsBatched	YES	DPCT1020 / DPCT1028

Function	Migration Support	Diagnostic Message
cublasDgelsBatched	YES	DPCT1020 / DPCT1028
cublasCgelsBatched	YES	DPCT1020 / DPCT1028
cublasZgelsBatched	YES	DPCT1020 / DPCT1028
cublasStpttr	NO	
cublasDtpttr	NO	
cublasCtpttr	NO	
cublasZtpttr	NO	
cublasStrttp	NO	
cublasDtrttp	NO	
cublasCtrttp	NO	
cublasZtrttp	NO	
cublasSgemmEx	YES	DPCT1020
cublasCgemmEx	YES	DPCT1020
cublasGemmEx	YES	DPCT1020
cublasGemmBatchedEx	YES	DPCT1020
cublasGemmStridedBatchedEx	YES	DPCT1020
cublasCsyrkEx	NO	
cublasCsyrk3mEx	NO	
cublasCherkEx	NO	
cublasCherk3mEx	NO	
cublasNrm2Ex	YES	DPCT1020
cublasAxpyEx	YES	DPCT1020
cublasDotEx	YES	DPCT1020
cublasScalEx	YES	DPCT1020
cublasLtCreate	YES	DPCT1026 / DPCT1027
cublasLtDestroy	YES	DPCT1026 / DPCT1027
cublasLtGetVersion	YES	DPCT1043
cublasLtGetCudartVersion	NO	
cublasLtGetProperty	NO	
cublasLtMatmul	YES	

Function	Migration Support	Diagnostic Message
cublasLtMatrixLayoutCreate	YES	
cublasLtMatrixLayoutGetAttribute	YES	
cublasLtMatrixLayoutSetAttribute	YES	
cublasLtMatrixLayoutDestroy	YES	
cublasLtMatrixTransform	YES	
cublasLtMatmulDescCreate	YES	
cublasLtMatmulDescDestroy	YES	
cublasLtMatmulDescSetAttribute	YES	
cublasLtMatmulDescGetAttribute	YES	
cublasLtMatrixTransformDescCreate	YES	
cublasLtMatrixTransformDescDestroy	YES	
cublasLtMatrixTransformDescSetAttribute	YES	
cublasLtMatrixTransformDescGetAttribute	YES	
cublasLtMatmulPreferenceCreate	YES	
cublasLtMatmulPreferenceDestroy	YES	
cublasLtMatmulPreferenceSetAttribute	YES	
cublasLtMatmulPreferenceGetAttribute	YES	
cublasLtMatmulAlgoCheck	NO	
cublasLtMatmulAlgoGetHeuristic	YES	
cublasLtMatmulAlgoGetIds	NO	
cublasLtMatmulAlgoInit	NO	
cublasLtMatmulAlgoCapGetAttribute	NO	
cublasLtMatmulAlgoConfigSetAttribute	NO	
cublasLtMatmulAlgoConfigGetAttribute	NO	
cublasXtCreate	NO	
cublasXtDestroy	NO	
cublasXtDeviceSelect	NO	
cublasXtSetBlockDim	NO	
cublasXtGetBlockDim	NO	

Function	Migration Support	Diagnostic Message
cublasXtSetCpuRoutine	NO	
cublasXtSetCpuRatio	NO	
cublasXtSetPinningMemMode	NO	
cublasXtGetPinningMemMode	NO	
cublasXtSgemm	NO	
cublasXtDgemm	NO	
cublasXtCgemm	NO	
cublasXtZgemm	NO	
cublasXtChemm	NO	
cublasXtZhemm	NO	
cublasXtSsymm	NO	
cublasXtDsymm	NO	
cublasXtCsymm	NO	
cublasXtZsymm	NO	
cublasXtSsyrk	NO	
cublasXtDsyrk	NO	
cublasXtCsyrk	NO	
cublasXtZsyrk	NO	
cublasXtSsyr2k	NO	
cublasXtDsyr2k	NO	
cublasXtCsyr2k	NO	
cublasXtZsyr2k	NO	
cublasXtSsyrkx	NO	
cublasXtDsyrkx	NO	
cublasXtCsyrkx	NO	
cublasXtZsyrkx	NO	
cublasXtCherk	NO	
cublasXtZherk	NO	
cublasXtCher2k	NO	
cublasXtZher2k	NO	

Function	Migration Support	Diagnostic Message
cublasXtCherkx	NO	
cublasXtZherkx	NO	
cublasXtStrsm	NO	
cublasXtDtrsm	NO	
cublasXtCtrsm	NO	
cublasXtZtrsm	NO	
cublasXtStrmm	NO	
cublasXtDtrmm	NO	
cublasXtCtrmm	NO	
cublasXtZtrmm	NO	
cublasXtSspmm	NO	
cublasXtDspmm	NO	
cublasXtCspmm	NO	
cublasXtZspmm	NO	
cublasInit	YES	DPCT1026 / DPCT1027
cublasShutdown	YES	DPCT1026 / DPCT1027
cublasGetError	YES	DPCT1026 / DPCT1027
cublasGetVersion	YES	
cublasAlloc	YES	
cublasFree	YES	
cublasSetKernelStream	YES	
cublasSnrm2	YES	
cublasDnrm2	YES	
cublasScnrm2	YES	
cublasDznrm2	YES	
cublasSdot	YES	
cublasDdot	YES	
cublasCdotu	YES	
cublasCdotc	YES	
cublasZdotu	YES	

Function	Migration Support	Diagnostic Message
cublasZdotc	YES	
cublasSscal	YES	
cublasDscal	YES	
cublasCscal	YES	
cublasZscal	YES	
cublasCsscal	YES	
cublasZdscal	YES	
cublasSaxpy	YES	
cublasDaxpy	YES	
cublasCaxpy	YES	
cublasZaxpy	YES	
cublasScopy	YES	
cublasDcopy	YES	
cublasCcopy	YES	
cublasZcopy	YES	
cublasSswap	YES	
cublasDswap	YES	
cublasCswap	YES	
cublasZswap	YES	
cublasIsamax	YES	
cublasIdamax	YES	
cublasIcamax	YES	
cublasIzamax	YES	
cublasIsamin	YES	
cublasIdamin	YES	
cublasIcamin	YES	
cublasIzamin	YES	
cublasSasum	YES	
cublasDasum	YES	
cublasScasum	YES	

Function	Migration Support	Diagnostic Message
cublasDzasum	YES	
cublasSrot	YES	
cublasDrot	YES	
cublasCrot	YES	
cublasZrot	YES	
cublasCsrot	YES	
cublasZdrot	YES	
cublasSrotg	YES	
cublasDrotg	YES	
cublasCrotg	NO	
cublasZrotg	NO	
cublasSrotm	YES	
cublasDrotm	YES	
cublasSrotmg	YES	
cublasDrotmg	YES	
cublasSgemv	YES	
cublasDgemv	YES	
cublasCgemv	YES	
cublasZgemv	YES	
cublasSgbmv	YES	
cublasDgbmv	YES	
cublasCgbmv	YES	
cublasZgbmv	YES	
cublasStrmv	YES	
cublasDtrmv	YES	
cublasCtrmv	YES	
cublasZtrmv	YES	
cublasStbmv	YES	
cublasDtbmv	YES	
cublasCtbmv	YES	

Function	Migration Support	Diagnostic Message
cublasZtbmv	YES	
cublasStpmv	YES	
cublasDtpmv	YES	
cublasCtpmv	YES	
cublasZtpmv	YES	
cublasStrsv	YES	
cublasDtrsv	YES	
cublasCtrsv	YES	
cublasZtrsv	YES	
cublasStpsv	YES	
cublasDtpsv	YES	
cublasCtpsv	YES	
cublasZtpsv	YES	
cublasStbsv	YES	
cublasDtbsv	YES	
cublasCtbsv	YES	
cublasZtbsv	YES	
cublasSsymv	YES	
cublasDsymv	YES	
cublasChemv	YES	
cublasZhemv	YES	
cublasSsbmv	YES	
cublasDsbmv	YES	
cublasChbmv	YES	
cublasZhbmv	YES	
cublasSspmv	YES	
cublasDspmv	YES	
cublasChpmv	YES	
cublasZhpmv	YES	
cublasSger	YES	

Function	Migration Support	Diagnostic Message
cublasDger	YES	
cublasCgeru	YES	
cublasCgerc	YES	
cublasZgeru	YES	
cublasZgerc	YES	
cublasSsyr	YES	
cublasDsyr	YES	
cublasCher	YES	
cublasZher	YES	
cublasSspr	YES	
cublasDspr	YES	
cublasChpr	YES	
cublasZhpr	YES	
cublasSsyr2	YES	
cublasDsyr2	YES	
cublasCher2	YES	
cublasZher2	YES	
cublasSspr2	YES	
cublasDspr2	YES	
cublasChpr2	YES	
cublasZhpr2	YES	
cublasSgemm	YES	
cublasDgemm	YES	
cublasCgemm	YES	
cublasZgemm	YES	
cublasSsyrk	YES	
cublasDsyrk	YES	
cublasCsyrk	YES	
cublasZsyrk	YES	
cublasCherk	YES	

Function	Migration Support	Diagnostic Message
cublasZherk	YES	
cublasSsyr2k	YES	
cublasDsyr2k	YES	
cublasCsyr2k	YES	
cublasZsyr2k	YES	
cublasCher2k	YES	
cublasZher2k	YES	
cublasSsymm	YES	
cublasDsymm	YES	
cublasCsymm	YES	
cublasZsymm	YES	
cublasChemm	YES	
cublasZhemm	YES	
cublasStrsm	YES	
cublasDtrsm	YES	
cublasCtrsm	YES	
cublasZtrsm	YES	
cublasStrmm	YES	
cublasDtrmm	YES	
cublasCtrmm	YES	
cublasZtrmm	YES	
cublasAsumEx	NO	
cublasCgemm3mBatched	NO	
cublasCgemm3mEx	NO	
cublasCopyEx	NO	
cublasDotcEx	YES	
cublasGetCudartVersion	NO	
cublasIamaxEx	NO	
cublasIaminEx	NO	
cublasRotEx	YES	

Function	Migration Support	Diagnostic Message
cublasRotgEx	NO	
cublasRotmEx	NO	
cublasRotmgEx	NO	
cublasSwapEx	NO	
cublasUint8gemmBias	NO	
cublasXerbla	NO	
cublasXtGetNumBoards	NO	
cublasXtMaxBoards	NO	
cublasGetSmCountTarget	NO	
cublasSetSmCountTarget	NO	
cublasGetStatusName	NO	
cublasGetStatusString	YES	DPCT1009
cublasLogCallback	NO	
cublasMigrateComputeType	NO	
cublasSetWorkspace	YES	DPCT1026 / DPCT1027
cublasSgemvBatched	NO	
cublasDgemvBatched	NO	
cublasCgemvBatched	NO	
cublasZgemvBatched	NO	
cublasSgemvStridedBatched	NO	
cublasDgemvStridedBatched	NO	
cublasCgemvStridedBatched	NO	
cublasZgemvStridedBatched	NO	
cublasHSHgemvBatched	NO	
cublasHSHgemvStridedBatched	NO	
cublasHSSgemvBatched	NO	
cublasHSSgemvStridedBatched	NO	
cublasTSSgemvBatched	NO	
cublasTSSgemvStridedBatched	NO	
cublasTSTgemvBatched	NO	

Function	Migration Support	Diagnostic Message
cublasTSTgemvStridedBatched	NO	
cublasLtGetStatusName	NO	
cublasLtGetStatusString	NO	
cublasLtLoggerForceDisable	NO	
cublasLtLoggerOpenFile	NO	
cublasLtLoggerSetCallback	NO	
cublasLtLoggerSetFile	NO	
cublasLtLoggerSetLevel	NO	
cublasLtLoggerSetMask	NO	
cublasLtMatmulDescInit	NO	
cublasLtMatmulPreferenceInit	NO	
cublasLtMatrixLayoutInit	NO	
cublasLtMatrixTransformDescInit	NO	
cublasSnrm2_64	YES	DPCT1020 / DPCT1034
cublasDnrm2_64	YES	DPCT1020 / DPCT1034
cublasScnrm2_64	YES	DPCT1020 / DPCT1034
cublasDznrm2_64	YES	DPCT1020 / DPCT1034
cublasSdot_64	YES	DPCT1020 / DPCT1034
cublasDdot_64	YES	DPCT1020 / DPCT1034
cublasCdotu_64	YES	DPCT1020 / DPCT1034
cublasCdotc_64	YES	DPCT1020 / DPCT1034
cublasZdotu_64	YES	DPCT1020 / DPCT1034
cublasZdotc_64	YES	DPCT1020 / DPCT1034
cublasSscal_64	YES	DPCT1020
cublasDscal_64	YES	DPCT1020
cublasCscal_64	YES	DPCT1020
cublasCsscal_64	YES	DPCT1020
cublasZscal_64	YES	DPCT1020
cublasZdscal_64	YES	DPCT1020
cublasSaxpy_64	YES	DPCT1020

Function	Migration Support	Diagnostic Message
cublasDaxpy_64	YES	DPCT1020
cublasCaxpy_64	YES	DPCT1020
cublasZaxpy_64	YES	DPCT1020
cublasScopy_64	YES	DPCT1020
cublasDcopy_64	YES	DPCT1020
cublasCcopy_64	YES	DPCT1020
cublasZcopy_64	YES	DPCT1020
cublasSswap_64	YES	DPCT1020
cublasDswap_64	YES	DPCT1020
cublasCswap_64	YES	DPCT1020
cublasZswap_64	YES	DPCT1020
cublasIsamax_64	YES	DPCT1020 / DPCT1034
cublasIdamax_64	YES	DPCT1020 / DPCT1034
cublasIcamax_64	YES	DPCT1020 / DPCT1034
cublasIzamax_64	YES	DPCT1020 / DPCT1034
cublasIsamin_64	YES	DPCT1020 / DPCT1034
cublasIdamin_64	YES	DPCT1020 / DPCT1034
cublasIcamin_64	YES	DPCT1020 / DPCT1034
cublasIzamin_64	YES	DPCT1020 / DPCT1034
cublasSasum_64	YES	DPCT1020 / DPCT1034
cublasDasum_64	YES	DPCT1020 / DPCT1034
cublasScasum_64	YES	DPCT1020 / DPCT1034
cublasDzasum_64	YES	DPCT1020 / DPCT1034
cublasSrot_64	YES	DPCT1020
cublasDrot_64	YES	DPCT1020
cublasCrot_64	YES	DPCT1020
cublasCsrot_64	YES	DPCT1020
cublasZrot_64	YES	DPCT1020
cublasZdrot_64	YES	DPCT1020
cublasSrotm_64	YES	DPCT1020 / DPCT1034

Function	Migration Support	Diagnostic Message
cublasDrotm_64	YES	DPCT1020 / DPCT1034
cublasSgemv_64	YES	DPCT1020
cublasDgemv_64	YES	DPCT1020
cublasCgemv_64	YES	DPCT1020
cublasZgemv_64	YES	DPCT1020
cublasSgbmv_64	YES	DPCT1020
cublasDgbmv_64	YES	DPCT1020
cublasCgbmv_64	YES	DPCT1020
cublasZgbmv_64	YES	DPCT1020
cublasStrmv_64	YES	DPCT1020
cublasDtrmv_64	YES	DPCT1020
cublasCtrmv_64	YES	DPCT1020
cublasZtrmv_64	YES	DPCT1020
cublasStbmv_64	YES	DPCT1020
cublasDtbmv_64	YES	DPCT1020
cublasCtbmv_64	YES	DPCT1020
cublasZtbmv_64	YES	DPCT1020
cublasStpmv_64	YES	DPCT1020
cublasDtpmv_64	YES	DPCT1020
cublasCtpmv_64	YES	DPCT1020
cublasZtpmv_64	YES	DPCT1020
cublasStrsv_64	YES	DPCT1020
cublasDtrsv_64	YES	DPCT1020
cublasCtrsv_64	YES	DPCT1020
cublasZtrsv_64	YES	DPCT1020
cublasStpsv_64	YES	DPCT1020
cublasDtpsv_64	YES	DPCT1020
cublasCtpsv_64	YES	DPCT1020
cublasZtpsv_64	YES	DPCT1020
cublasStbsv_64	NO	

Function	Migration Support	Diagnostic Message
cublasDtbsv_64	NO	
cublasCtbsv_64	NO	
cublasZtbsv_64	NO	
cublasSsymv_64	NO	
cublasDsymv_64	NO	
cublasCsymv_64	NO	
cublasZsymv_64	NO	
cublasChemv_64	NO	
cublasZhemv_64	NO	
cublasSsbmv_64	NO	
cublasDsbmv_64	NO	
cublasChbmv_64	NO	
cublasZhbmv_64	NO	
cublasSspmv_64	NO	
cublasDspmv_64	NO	
cublasChpmv_64	NO	
cublasZhpmv_64	NO	
cublasSger_64	NO	
cublasDger_64	NO	
cublasCgeru_64	NO	
cublasCgerc_64	NO	
cublasZgeru_64	NO	
cublasZgerc_64	NO	
cublasSsyr_64	NO	
cublasDsyr_64	NO	
cublasCsyr_64	NO	
cublasZsyr_64	NO	
cublasCher_64	NO	
cublasZher_64	NO	
cublasSspr_64	NO	

Function	Migration Support	Diagnostic Message
cublasDspr_64	NO	
cublasChpr_64	NO	
cublasZhpr_64	NO	
cublasSsyr2_64	NO	
cublasDsyr2_64	NO	
cublasCsyr2_64	NO	
cublasZsyr2_64	NO	
cublasCher2_64	NO	
cublasZher2_64	NO	
cublasSspr2_64	NO	
cublasDspr2_64	NO	
cublasChpr2_64	NO	
cublasZhpr2_64	NO	
cublasSgemm_64	YES	DPCT1020
cublasDgemm_64	YES	DPCT1020
cublasCgemm_64	YES	DPCT1020
cublasZgemm_64	YES	DPCT1020
cublasSsyrk_64	YES	DPCT1020
cublasDsyrk_64	YES	DPCT1020
cublasCsyrk_64	YES	DPCT1020
cublasZsyrk_64	YES	DPCT1020
cublasCherk_64	YES	DPCT1020
cublasZherk_64	YES	DPCT1020
cublasSsyr2k_64	YES	DPCT1020
cublasDsyr2k_64	YES	DPCT1020
cublasCsyr2k_64	YES	DPCT1020
cublasZsyr2k_64	YES	DPCT1020
cublasCher2k_64	YES	DPCT1020
cublasZher2k_64	YES	DPCT1020
cublasSsymm_64	YES	DPCT1020

Function	Migration Support	Diagnostic Message
cublasDsymm_64	YES	DPCT1020
cublasCsymm_64	YES	DPCT1020
cublasZsymm_64	YES	DPCT1020
cublasChemm_64	YES	DPCT1020
cublasZhemm_64	YES	DPCT1020
cublasStrsm_64	YES	DPCT1020
cublasDtrsm_64	YES	DPCT1020
cublasCtrsm_64	YES	DPCT1020
cublasZtrsm_64	YES	DPCT1020
cublasStrmm_64	YES	DPCT1020
cublasDtrmm_64	YES	DPCT1020
cublasCtrmm_64	YES	DPCT1020
cublasZtrmm_64	YES	DPCT1020
cublasSgeam_64	YES	DPCT1020
cublasDgeam_64	YES	DPCT1020
cublasCgeam_64	YES	DPCT1020
cublasZgeam_64	YES	DPCT1020
cublasSdgmm_64	YES	DPCT1020
cublasDdgmm_64	YES	DPCT1020
cublasCdgmm_64	YES	DPCT1020
cublasZdgmm_64	YES	DPCT1020
cublasSetVector_64	YES	DPCT1018 / DPCT1020
cublasGetVector_64	YES	DPCT1018 / DPCT1020
cublasSetMatrix_64	YES	DPCT1018 / DPCT1020
cublasGetMatrix_64	YES	DPCT1018 / DPCT1020
cublasSetVectorAsync_64	YES	DPCT1018 / DPCT1020
cublasGetVectorAsync_64	YES	DPCT1018 / DPCT1020
cublasSetMatrixAsync_64	YES	DPCT1018 / DPCT1020
cublasGetMatrixAsync_64	YES	DPCT1018 / DPCT1020
cublasNrm2Ex_64	NO	

Function	Migration Support	Diagnostic Message
cublasDotEx_64	NO	
cublasDotcEx_64	NO	
cublasScalEx_64	NO	
cublasAxpyEx_64	NO	
cublasCopyEx_64	NO	
cublasSwapEx_64	NO	
cublasIamaxEx_64	NO	
cublasIaminEx_64	NO	
cublasAsumEx_64	NO	
cublasRotEx_64	NO	
cublasRotmEx_64	NO	
cublasSgemvBatched_64	NO	
cublasDgemvBatched_64	NO	
cublasCgemvBatched_64	NO	
cublasZgemvBatched_64	NO	
cublasHSHgemvBatched_64	NO	
cublasHSSgemvBatched_64	NO	
cublasTSTgemvBatched_64	NO	
cublasTSSgemvBatched_64	NO	
cublasSgemvStridedBatched_64	NO	
cublasDgemvStridedBatched_64	NO	
cublasCgemvStridedBatched_64	NO	
cublasZgemvStridedBatched_64	NO	
cublasHSHgemvStridedBatched_64	NO	
cublasHSSgemvStridedBatched_64	NO	
cublasTSTgemvStridedBatched_64	NO	
cublasTSSgemvStridedBatched_64	NO	
cublasCgemm3m_64	YES	DPCT1020
cublasCgemm3mEx_64	NO	
cublasZgemm3m_64	YES	DPCT1020

Function	Migration Support	Diagnostic Message
cublasHgemm_64	YES	DPCT1020
cublasSgemmEx_64	NO	
cublasGemmEx_64	NO	
cublasCgemmEx_64	NO	
cublasCsyrkEx_64	NO	
cublasCsyrk3mEx_64	NO	
cublasCherkEx_64	NO	
cublasCherk3mEx_64	NO	
cublasSsyrkx_64	YES	DPCT1020
cublasDsyrkx_64	YES	DPCT1020
cublasCsyrkx_64	YES	DPCT1020
cublasZsyrkx_64	YES	DPCT1020
cublasCherkx_64	YES	DPCT1020
cublasZherkx_64	YES	DPCT1020
cublasHgemmBatched_64	NO	
cublasSgemmBatched_64	NO	
cublasDgemmBatched_64	NO	
cublasCgemmBatched_64	NO	
cublasCgemm3mBatched_64	NO	
cublasZgemmBatched_64	NO	
cublasHgemmStridedBatched_64	NO	
cublasSgemmStridedBatched_64	NO	
cublasDgemmStridedBatched_64	NO	
cublasCgemmStridedBatched_64	NO	
cublasCgemm3mStridedBatched_64	NO	
cublasZgemmStridedBatched_64	NO	
cublasGemmBatchedEx_64	NO	
cublasGemmStridedBatchedEx_64	NO	
cublasSgemmGroupedBatched	NO	
cublasSgemmGroupedBatched_64	NO	

Function	Migration Support	Diagnostic Message
cublasDgemmGroupedBatched	NO	
cublasDgemmGroupedBatched_64	NO	
cublasGemmGroupedBatchedEx	NO	
cublasGemmGroupedBatchedEx_64	NO	
cublasStrsmBatched_64	NO	
cublasDtrsmBatched_64	NO	
cublasCtrsmBatched_64	NO	
cublasZtrsmBatched_64	NO	
cublasLtHeuristicsCacheGetCapacity	NO	
cublasLtHeuristicsCacheSetCapacity	NO	
cublasLtDisableCpuInstructionsSetMask	NO	

cuDNN API

Function	Migration Support	Diagnostic Message
cudnnActivationForward	YES	
cudnnAddTensor	YES	
cudnnBatchNormalizationForwardInference	YES	
cudnnCopyAlgorithmDescriptor	NO	
cudnnCreate	YES	
cudnnCreateActivationDescriptor	YES	
cudnnCreateAlgorithmDescriptor	NO	
cudnnCreateAlgorithmPerformance	NO	
cudnnCreateDropoutDescriptor	YES	
cudnnCreateFilterDescriptor	YES	
cudnnCreateLRNDescriptor	YES	
cudnnCreateOpTensorDescriptor	YES	
cudnnCreatePoolingDescriptor	YES	
cudnnCreateReduceTensorDescriptor	YES	
cudnnCreateSpatialTransformerDescriptor	NO	

Function	Migration Support	Diagnostic Message
cudnnCreateTensorDescriptor	YES	
cudnnCreateTensorTransformDescriptor	NO	
cudnnDeriveBNTensorDescriptor	YES	
cudnnDeriveNormTensorDescriptor	YES	
cudnnDestroy	YES	
cudnnDestroyActivationDescriptor	YES	
cudnnDestroyAlgorithmDescriptor	NO	
cudnnDestroyAlgorithmPerformance	NO	
cudnnDestroyDropoutDescriptor	YES	
cudnnDestroyFilterDescriptor	YES	
cudnnDestroyLRNDescriptor	YES	
cudnnDestroyOpTensorDescriptor	YES	
cudnnDestroyPoolingDescriptor	YES	
cudnnDestroyReduceTensorDescriptor	YES	
cudnnDestroySpatialTransformerDescriptor	NO	
cudnnDestroyTensorDescriptor	YES	
cudnnDestroyTensorTransformDescriptor	NO	
cudnnDivisiveNormalizationForward	NO	
cudnnDropoutForward	YES	
cudnnDropoutGetReserveSpaceSize	YES	
cudnnDropoutGetStatesSize	YES	
cudnnGetActivationDescriptor	YES	
cudnnGetActivationDescriptorSwishBeta	YES	
cudnnGetAlgorithmDescriptor	NO	
cudnnGetAlgorithmPerformance	NO	
cudnnGetAlgorithmSpaceSize	NO	
cudnnGetCallback	NO	
cudnnGetCudartVersion	YES	
cudnnGetDropoutDescriptor	YES	
cudnnGetErrorString	YES	

Function	Migration Support	Diagnostic Message
cudnnGetFilter4dDescriptor	YES	
cudnnGetFilterNdDescriptor	YES	
cudnnGetFilterSizeInBytes	YES	
cudnnGetLRNDescriptor	YES	
cudnnGetOpTensorDescriptor	YES	
cudnnGetPooling2dDescriptor	YES	
cudnnGetPooling2dForwardOutputDim	YES	
cudnnGetPoolingNdDescriptor	YES	
cudnnGetPoolingNdForwardOutputDim	YES	
cudnnGetProperty	NO	
cudnnGetReduceTensorDescriptor	YES	
cudnnGetReductionIndicesSize	NO	
cudnnGetReductionWorkspaceSize	YES	
cudnnGetStream	YES	
cudnnGetTensor4dDescriptor	YES	
cudnnGetTensorNdDescriptor	YES	
cudnnGetTensorSizeInBytes	YES	
cudnnGetTensorTransformDescriptor	NO	
cudnnGetVersion	YES	
cudnnInitTransformDest	NO	
cudnnLRNCrossChannelForward	YES	
cudnnNormalizationForwardInference	YES	
cudnnOpsInferVersionCheck	NO	
cudnnOpTensor	YES	
cudnnPoolingForward	YES	
cudnnQueryRuntimeError	NO	
cudnnReduceTensor	YES	
cudnnRestoreAlgorithm	NO	
cudnnRestoreDropoutDescriptor	YES	
cudnnSaveAlgorithm	NO	

Function	Migration Support	Diagnostic Message
cudnnScaleTensor	YES	
cudnnSetActivationDescriptor	YES	
cudnnSetActivationDescriptorSwishBeta	YES	
cudnnSetAlgorithmDescriptor	NO	
cudnnSetAlgorithmPerformance	NO	
cudnnSetCallback	NO	
cudnnSetDropoutDescriptor	YES	
cudnnSetFilter4dDescriptor	YES	
cudnnSetFilterNdDescriptor	YES	
cudnnSetLRNDescriptor	YES	
cudnnSetOpTensorDescriptor	YES	
cudnnSetPooling2dDescriptor	YES	
cudnnSetPoolingNdDescriptor	YES	
cudnnSetReduceTensorDescriptor	YES	
cudnnSetSpatialTransformerNdDescriptor	NO	
cudnnSetStream	YES	
cudnnSetTensor	YES	
cudnnSetTensor4dDescriptor	YES	
cudnnSetTensor4dDescriptorEx	YES	
cudnnSetTensorNdDescriptor	YES	
cudnnSetTensorNdDescriptorEx	YES	
cudnnSetTensorTransformDescriptor	NO	
cudnnSoftmaxForward	YES	
cudnnSpatialTfGridGeneratorForward	NO	
cudnnSpatialTfSamplerForward	NO	
cudnnTransformFilter	NO	
cudnnTransformTensor	YES	
cudnnTransformTensorEx	NO	
cudnnActivationBackward	YES	
cudnnBatchNormalizationBackward	YES	

Function	Migration Support	Diagnostic Message
cudnnBatchNormalizationBackwardEx	YES	
cudnnBatchNormalizationForwardTraining	YES	
cudnnBatchNormalizationForwardTrainingEx	YES	
cudnnDivisiveNormalizationBackward	NO	
cudnnDropoutBackward	YES	
cudnnGetBatchNormalizationBackwardExWorkspaceSize	YES	
cudnnGetBatchNormalizationForwardTrainingExWorkspaceSize	YES	
cudnnGetBatchNormalizationTrainingExReserveSpaceSize	YES	
cudnnGetNormalizationBackwardWorkspaceSize	YES	
cudnnGetNormalizationForwardTrainingWorkspaceSize	YES	
cudnnGetNormalizationTrainingReserveSpaceSize	YES	
cudnnLRNCrossChannelBackward	YES	DPCT1097
cudnnNormalizationBackward	YES	
cudnnNormalizationForwardTraining	YES	
cudnnOpsTrainVersionCheck	NO	
cudnnPoolingBackward	YES	DPCT1097
cudnnSoftmaxBackward	YES	
cudnnSpatialTfGridGeneratorBackward	NO	
cudnnSpatialTfSamplerBackward	NO	
cudnnBackendCreateDescriptor	NO	
cudnnBackendDestroyDescriptor	NO	
cudnnBackendExecute	NO	
cudnnBackendFinalize	NO	
cudnnBackendGetAttribute	NO	
cudnnBackendInitialize	NO	
cudnnBackendSetAttribute	NO	

Function	Migration Support	Diagnostic Message
cudnnConvolutionBackwardData	YES	
cudnnConvolutionBiasActivationForward	YES	
cudnnConvolutionForward	YES	
cudnnCreateConvolutionDescriptor	YES	
cudnnDestroyConvolutionDescriptor	YES	
cudnnFindConvolutionBackwardDataAlgorithm	YES	
cudnnFindConvolutionBackwardDataAlgorithmEx	NO	
cudnnFindConvolutionForwardAlgorithm	YES	
cudnnFindConvolutionForwardAlgorithmEx	NO	
cudnnGetConvolution2dDescriptor	YES	
cudnnGetConvolution2dForwardOutputDim	YES	
cudnnGetConvolutionBackwardDataAlgorithmMaxCount	YES	
cudnnGetConvolutionBackwardDataAlgorithm_v7	YES	
cudnnGetConvolutionBackwardDataAlgorithm	YES	
cudnnGetConvolutionBackwardDataWorkspaceSize	YES	
cudnnGetConvolutionForwardAlgorithmMaxCount	YES	
cudnnGetConvolutionForwardAlgorithm_v7	YES	
cudnnGetConvolutionForwardAlgorithm	YES	
cudnnGetConvolutionForwardWorkspaceSize	YES	
cudnnGetConvolutionGroupCount	YES	
cudnnGetConvolutionMathType	NO	
cudnnGetConvolutionNdDescriptor	YES	
cudnnGetConvolutionNdForwardOutputDim	YES	

Function	Migration Support	Diagnostic Message
cudnnGetConvolutionReorderType	NO	
cudnnGetFoldedConvBackwardDataDescriptors	NO	
cudnnIm2Col	NO	
cudnnReorderFilterAndBias	NO	
cudnnSetConvolution2dDescriptor	YES	
cudnnSetConvolutionGroupCount	YES	
cudnnSetConvolutionMathType	YES	
cudnnSetConvolutionNdDescriptor	YES	
cudnnSetConvolutionReorderType	NO	
cudnnConvolutionBackwardBias	YES	
cudnnConvolutionBackwardFilter	YES	
cudnnCreateFusedOpsConstParamPack	NO	
cudnnCreateFusedOpsPlan	NO	
cudnnCreateFusedOpsVariantParamPack	NO	
cudnnDestroyFusedOpsConstParamPack	NO	
cudnnDestroyFusedOpsPlan	NO	
cudnnDestroyFusedOpsVariantParamPack	NO	
cudnnFindConvolutionBackwardFilterAlgorithm	YES	
cudnnFindConvolutionBackwardFilterAlgorithmEx	NO	
cudnnFusedOpsExecute	NO	
cudnnGetConvolutionBackwardFilterAlgorithmMaxCount	YES	
cudnnGetConvolutionBackwardFilterAlgorithm_v7	YES	
cudnnGetConvolutionBackwardFilterAlgorithm	YES	
cudnnGetConvolutionBackwardFilterWorkspaceSize	YES	
cudnnGetFusedOpsConstParamPackAttribute	NO	

Function	Migration Support	Diagnostic Message
cudnnGetFusedOpsVariantParamPackAttribute	NO	
cudnnMakeFusedOpsPlan	NO	
cudnnSetFusedOpsConstParamPackAttribute	NO	
cudnnSetFusedOpsVariantParamPackAttribute	NO	
cudnnAdvInferVersionCheck	NO	
cudnnBuildRNNDynamic	NO	
cudnnCreateAttnDescriptor	NO	
cudnnCreatePersistentRNNPlan	NO	
cudnnCreateRNNDataDescriptor	YES	
cudnnCreateRNNDescriptor	YES	
cudnnCreateSeqDataDescriptor	NO	
cudnnDestroyAttnDescriptor	NO	
cudnnDestroyPersistentRNNPlan	NO	
cudnnDestroyRNNDataDescriptor	YES	
cudnnDestroyRNNDescriptor	YES	
cudnnDestroySeqDataDescriptor	NO	
cudnnFindRNNForwardInferenceAlgorithmEx	NO	
cudnnGetAttnDescriptor	NO	
cudnnGetMultiHeadAttnBuffers	NO	
cudnnGetMultiHeadAttnWeights	NO	
cudnnGetRNNBackwardDataAlgorithmMaxCount	NO	
cudnnGetRNNBiasMode	NO	
cudnnGetRNNDataDescriptor	YES	
cudnnGetRNNDescriptor_v6	NO	
cudnnGetRNNDescriptor_v8	YES	
cudnnGetRNNForwardInferenceAlgorithmMaxCount	NO	

Function	Migration Support	Diagnostic Message
cudnnGetRNNForwardTrainingAlgorithmMaxCount	NO	
cudnnGetRNNLinLayerBiasParams	NO	
cudnnGetRNNLinLayerMatrixParams	NO	
cudnnGetRNNMatrixMathType	NO	
cudnnGetRNNPaddingMode	NO	
cudnnGetRNNParamsSize	NO	
cudnnGetRNNProjectionLayers	NO	
cudnnGetRNNTempSpaceSizes	YES	
cudnnGetRNNWeightParams	NO	
cudnnGetRNNWeightSpaceSize	YES	
cudnnGetRNNWorkspaceSize	NO	
cudnnGetSeqDataDescriptor	NO	
cudnnMultiHeadAttnForward	NO	
cudnnRNNForward	YES	
cudnnRNNForwardInference	NO	
cudnnRNNForwardInferenceEx	NO	
cudnnRNNGetClip	NO	
cudnnRNNGetClip_v8	NO	
cudnnRNNSetClip	NO	
cudnnRNNSetClip_v8	NO	
cudnnSetAttnDescriptor	NO	
cudnnSetPersistentRNNPlan	NO	
cudnnSetRNNAlgorithmDescriptor	NO	
cudnnSetRNNBiasMode	NO	
cudnnSetRNNDataDescriptor	YES	
cudnnSetRNNDescriptor_v6	NO	
cudnnSetRNNDescriptor_v8	YES	
cudnnSetRNNMatrixMathType	NO	
cudnnSetRNNPaddingMode	NO	
cudnnSetRNNProjectionLayers	NO	

Function	Migration Support	Diagnostic Message
cudnnSetSeqDataDescriptor	NO	
cudnnAdvTrainVersionCheck	NO	
cudnnCreateCTCLossDescriptor	NO	
cudnnCTCLoss	NO	
cudnnCTCLoss_v8	NO	
cudnnDestroyCTCLossDescriptor	NO	
cudnnFindRNNBackwardDataAlgorithmEx	NO	
cudnnFindRNNBackwardWeightsAlgorithmEx	NO	
cudnnFindRNNForwardTrainingAlgorithmEx	NO	
cudnnGetCTCLossDescriptor	NO	
cudnnGetCTCLossDescriptorEx	NO	
cudnnGetCTCLossDescriptor_v8	NO	
cudnnGetCTCLossWorkspaceSize	NO	
cudnnGetCTCLossWorkspaceSize_v8	NO	
cudnnGetRNNTrainingReserveSize	NO	
cudnnMultiHeadAttnBackwardData	NO	
cudnnMultiHeadAttnBackwardWeights	NO	
cudnnRNNBackwardData	NO	
cudnnRNNBackwardData_v8	YES	
cudnnRNNBackwardDataEx	NO	
cudnnRNNBackwardWeights	NO	
cudnnRNNBackwardWeights_v8	YES	
cudnnRNNBackwardWeightsEx	NO	
cudnnRNNForwardTraining	NO	
cudnnRNNForwardTrainingEx	NO	
cudnnSetCTCLossDescriptor	NO	
cudnnSetCTCLossDescriptorEx	NO	
cudnnSetCTCLossDescriptor_v8	NO	
cudnnGetCTCLossDescriptor_v9	NO	

Function	Migration Support	Diagnostic Message
cudnnRNNGetClip_v9	NO	
cudnnRNNSetClip_v9	NO	
cudnnSetCTCLossDescriptor_v9	NO	
cudnnGetLastErrorString	NO	

cuFFT API

Function	Migration Support	Diagnostic Message
cufftPlan1d	YES	
cufftPlan2d	YES	
cufftPlan3d	YES	
cufftPlanMany	YES	
cufftXtMakePlanMany	YES	DPCT1099 / DPCT1100
cufftExecC2C	YES	
cufftExecZ2Z	YES	
cufftExecR2C	YES	
cufftExecD2Z	YES	
cufftExecC2R	YES	
cufftExecZ2D	YES	
cufftCreate	YES	
cufftXtSetGPUs	NO	
cufftMakePlan1d	YES	DPCT1099 / DPCT1100
cufftMakePlan2d	YES	DPCT1099 / DPCT1100
cufftMakePlan3d	YES	DPCT1099 / DPCT1100
cufftMakePlanMany	YES	DPCT1099 / DPCT1100
cufftGetSize1d	YES	DPCT1099 / DPCT1100
cufftGetSize2d	YES	DPCT1099 / DPCT1100
cufftGetSize3d	YES	DPCT1099 / DPCT1100
cufftGetSizeMany	YES	DPCT1099 / DPCT1100
cufftGetSize	YES	DPCT1100
cufftXtSetWorkArea	NO	

Function	Migration Support	Diagnostic Message
cufftXtMalloc	NO	
cufftXtMemcpy	NO	
cufftXtClearCallback	NO	
cufftXtExecDescriptorC2R	NO	
cufftXtExecDescriptorD2Z	NO	
cufftXtExecDescriptorR2C	NO	
cufftXtExecDescriptorZ2D	NO	
cufftXtExecDescriptorC2C	NO	
cufftXtExecDescriptorZ2Z	NO	
cufftXtQueryPlan	NO	
cufftXtSetCallback	NO	
cufftXtSetCallbackSharedSize	NO	
cufftXtFree	NO	
cufftDestroy	YES	
cufftSetAutoAllocation	YES	DPCT1100
cufftEstimate1d	YES	DPCT1099 / DPCT1100
cufftEstimate2d	YES	DPCT1099 / DPCT1100
cufftEstimate3d	YES	DPCT1099 / DPCT1100
cufftEstimateMany	YES	DPCT1099 / DPCT1100
cufftXtSetWorkAreaPolicy	NO	
cufftMakePlanMany64	YES	DPCT1099 / DPCT1100
cufftGetSizeMany64	YES	DPCT1099 / DPCT1100
cufftXtGetSizeMany	NO	
cufftSetWorkArea	YES	DPCT1100
cufftXtExec	YES	
cufftXtExecDescriptor	NO	
cufftSetStream	YES	
cufftGetVersion	YES	
cufftGetProperty	YES	
cufftSetPlanPropertyInt64	NO	

Function	Migration Support	Diagnostic Message
cufftGetPlanPropertyInt64	NO	
cufftResetPlanProperty	NO	

cuRAND API

Function	Migration Support	Diagnostic Message
curandCreateGenerator	YES	DPCT1028 / DPCT1032 / DPCT1033
curandCreateGeneratorHost	YES	DPCT1028 / DPCT1032 / DPCT1033
curandCreatePoissonDistribution	NO	
curandDestroyDistribution	NO	
curandDestroyGenerator	YES	
curandGenerate	YES	
curandGenerateLogNormal	YES	
curandGenerateLogNormalDouble	YES	
curandGenerateLongLong	YES	
curandGenerateNormal	YES	
curandGenerateNormalDouble	YES	
curandGeneratePoisson	YES	
curandGenerateSeeds	NO	
curandGenerateUniform	YES	
curandGenerateUniformDouble	YES	
curandGetDirectionVectors32	NO	
curandGetDirectionVectors64	NO	
curandGetProperty	NO	
curandGetScrambleConstants32	NO	
curandGetScrambleConstants64	NO	
curandGetVersion	NO	
curandSetGeneratorOffset	YES	
curandSetGeneratorOrdering	YES	
curandSetPseudoRandomGeneratorSeed	YES	

Function	Migration Support	Diagnostic Message
curandSetQuasiRandomGeneratorDimensions	YES	
curandSetStream	YES	
curandMakeMTGP32KernelState	NO	
curand	YES	
curand4	YES	
curand_init	YES	DPCT1028 / DPCT1105
curand_log_normal	YES	
curand_log_normal2	YES	
curand_log_normal2_double	YES	
curand_log_normal4	YES	
curand_log_normal_double	YES	
curand_normal	YES	
curand_normal2	YES	
curand_normal2_double	YES	
curand_normal4	YES	
curand_normal_double	YES	
curand_poisson	YES	
curand_poisson4	YES	
curand_uniform	YES	
curand_uniform2_double	YES	
curand_uniform4	YES	
curand_uniform_double	YES	
skipahead	YES	
skipahead_sequence	YES	DPCT1026 / DPCT1027
skipahead_subsequence	YES	
curand_uniform4_double	YES	
curand_normal4_double	YES	
curand_log_normal4_double	YES	
curand_discrete	NO	
curand_discrete4	NO	

Function	Migration Support	Diagnostic Message
curandMakeMTGP32Constants	NO	
curand_mtgp32_single	NO	
curand_mtgp32_single_specific	NO	
curand_mtgp32_specific	NO	
__curand_umul	NO	
curand_Philox4x32_10	NO	

cuSOLVER API

Function	Migration Support	Diagnostic Message
cusolverDnCreate	YES	
cusolverDnDestroy	YES	
cusolverDnSetStream	YES	
cusolverDnGetStream	YES	
cusolverDnCreateSyevjInfo	YES	DPCT1026 / DPCT1027
cusolverDnDestroySyevjInfo	YES	DPCT1026 / DPCT1027
cusolverDnXsyevjSetTolerance	NO	
cusolverDnXsyevjSetMaxSweeps	NO	
cusolverDnXsyevjSetSortEig	NO	
cusolverDnXsyevjGetResidual	NO	
cusolverDnXsyevjGetSweeps	NO	
cusolverDnCreateGesvdjInfo	YES	DPCT1026 / DPCT1027
cusolverDnDestroyGesvdjInfo	YES	DPCT1026 / DPCT1027
cusolverDnXgesvdjSetTolerance	NO	
cusolverDnXgesvdjSetMaxSweeps	NO	
cusolverDnXgesvdjSetSortEig	NO	
cusolverDnXgesvdjGetResidual	NO	
cusolverDnXgesvdjGetSweeps	NO	
cusolverDnSpotrf_bufferSize	YES	
cusolverDnDpotrf_bufferSize	YES	
cusolverDnCpotrf_bufferSize	YES	

Function	Migration Support	Diagnostic Message
cusolverDnZpotrf_bufferSize	YES	
cusolverDnSpotrf	YES	
cusolverDnDpotrf	YES	
cusolverDnCpotrf	YES	
cusolverDnZpotrf	YES	
cusolverDnSpotrs	YES	
cusolverDnDpotrs	YES	
cusolverDnCpotrs	YES	
cusolverDnZpotrs	YES	
cusolverDnSpotri_bufferSize	YES	
cusolverDnDpotri_bufferSize	YES	
cusolverDnCpotri_bufferSize	YES	
cusolverDnZpotri_bufferSize	YES	
cusolverDnSpotri	YES	
cusolverDnDpotri	YES	
cusolverDnCpotri	YES	
cusolverDnZpotri	YES	
cusolverDnSgetrf_bufferSize	YES	
cusolverDnDgetrf_bufferSize	YES	
cusolverDnCgetrf_bufferSize	YES	
cusolverDnZgetrf_bufferSize	YES	
cusolverDnSgetrf	YES	DPCT1047
cusolverDnDgetrf	YES	DPCT1047
cusolverDnCgetrf	YES	DPCT1047
cusolverDnZgetrf	YES	DPCT1047
cusolverDnSgetrs	YES	
cusolverDnDgetrs	YES	
cusolverDnCgetrs	YES	
cusolverDnZgetrs	YES	
cusolverDnSgeqrf_bufferSize	YES	

Function	Migration Support	Diagnostic Message
cusolverDnDgeqrf_bufferSize	YES	
cusolverDnCgeqrf_bufferSize	YES	
cusolverDnZgeqrf_bufferSize	YES	
cusolverDnSgeqrf	YES	
cusolverDnDgeqrf	YES	
cusolverDnCgeqrf	YES	
cusolverDnZgeqrf	YES	
cusolverDnSormqr_bufferSize	YES	
cusolverDnDormqr_bufferSize	YES	
cusolverDnCunmqr_bufferSize	YES	
cusolverDnZunmqr_bufferSize	YES	
cusolverDnSormqr	YES	
cusolverDnDormqr	YES	
cusolverDnCunmqr	YES	
cusolverDnZunmqr	YES	
cusolverDnSorgqr_bufferSize	YES	
cusolverDnDorgqr_bufferSize	YES	
cusolverDnCungqr_bufferSize	YES	
cusolverDnZungqr_bufferSize	YES	
cusolverDnSorgqr	YES	
cusolverDnDorgqr	YES	
cusolverDnCungqr	YES	
cusolverDnZungqr	YES	
cusolverDnSsytrf_bufferSize	YES	
cusolverDnDsytrf_bufferSize	YES	
cusolverDnCsytrf_bufferSize	YES	
cusolverDnZsytrf_bufferSize	YES	
cusolverDnSsytrf	YES	
cusolverDnDsytrf	YES	
cusolverDnCsytrf	YES	

Function	Migration Support	Diagnostic Message
cusolverDnZsytrf	YES	
cusolverDnSpotrfBatched	YES	
cusolverDnDpotrfBatched	YES	
cusolverDnCpotrfBatched	YES	
cusolverDnZpotrfBatched	YES	
cusolverDnSpotrsBatched	YES	
cusolverDnDpotrsBatched	YES	
cusolverDnCpotrsBatched	YES	
cusolverDnZpotrsBatched	YES	
cusolverDnSgebrd_bufferSize	YES	
cusolverDnDgebrd_bufferSize	YES	
cusolverDnCgebrd_bufferSize	YES	
cusolverDnZgebrd_bufferSize	YES	
cusolverDnSgebrd	YES	
cusolverDnDgebrd	YES	
cusolverDnCgebrd	YES	
cusolverDnZgebrd	YES	
cusolverDnSorgbr_bufferSize	YES	
cusolverDnDorgbr_bufferSize	YES	
cusolverDnCungbr_bufferSize	YES	
cusolverDnZungbr_bufferSize	YES	
cusolverDnSorgbr	YES	
cusolverDnDorgbr	YES	
cusolverDnCungbr	YES	
cusolverDnZungbr	YES	
cusolverDnSsytrd_bufferSize	YES	
cusolverDnDsytrd_bufferSize	YES	
cusolverDnChetrd_bufferSize	YES	
cusolverDnZhetrd_bufferSize	YES	
cusolverDnSsytrd	YES	

Function	Migration Support	Diagnostic Message
cusolverDnDsytrd	YES	
cusolverDnChetrd	YES	
cusolverDnZhetrd	YES	
cusolverDnSormtr_bufferSize	YES	
cusolverDnDormtr_bufferSize	YES	
cusolverDnCunmtr_bufferSize	YES	
cusolverDnZunmtr_bufferSize	YES	
cusolverDnSormtr	YES	
cusolverDnDormtr	YES	
cusolverDnCunmtr	YES	
cusolverDnZunmtr	YES	
cusolverDnSorgtr_bufferSize	YES	
cusolverDnDorgtr_bufferSize	YES	
cusolverDnCungtr_bufferSize	YES	
cusolverDnZungtr_bufferSize	YES	
cusolverDnSorgtr	YES	
cusolverDnDorgtr	YES	
cusolverDnCungtr	YES	
cusolverDnZungtr	YES	
cusolverDnSgesvd_bufferSize	YES	
cusolverDnDgesvd_bufferSize	YES	
cusolverDnCgesvd_bufferSize	YES	
cusolverDnZgesvd_bufferSize	YES	
cusolverDnSgesvd	YES	
cusolverDnDgesvd	YES	
cusolverDnCgesvd	YES	
cusolverDnZgesvd	YES	
cusolverDnSgesvdj_bufferSize	YES	
cusolverDnDgesvdj_bufferSize	YES	
cusolverDnCgesvdj_bufferSize	YES	

Function	Migration Support	Diagnostic Message
cusolverDnZgesvdj_bufferSize	YES	
cusolverDnSgesvdj	YES	
cusolverDnDgesvdj	YES	
cusolverDnCgesvdj	YES	
cusolverDnZgesvdj	YES	
cusolverDnSgesvdjBatched_bufferSize	NO	
cusolverDnDgesvdjBatched_bufferSize	NO	
cusolverDnCgesvdjBatched_bufferSize	NO	
cusolverDnZgesvdjBatched_bufferSize	NO	
cusolverDnSgesvdjBatched	NO	
cusolverDnDgesvdjBatched	NO	
cusolverDnCgesvdjBatched	NO	
cusolverDnZgesvdjBatched	NO	
cusolverDnSgesvdaStridedBatched_bufferSize	NO	
cusolverDnDgesvdaStridedBatched_bufferSize	NO	
cusolverDnCgesvdaStridedBatched_bufferSize	NO	
cusolverDnZgesvdaStridedBatched_bufferSize	NO	
cusolverDnSgesvdaStridedBatched	NO	
cusolverDnDgesvdaStridedBatched	NO	
cusolverDnCgesvdaStridedBatched	NO	
cusolverDnZgesvdaStridedBatched	NO	
cusolverDnSsyevd_bufferSize	YES	
cusolverDnDsyevd_bufferSize	YES	
cusolverDnCheevd_bufferSize	YES	
cusolverDnZheevd_bufferSize	YES	
cusolverDnSsyevd	YES	
cusolverDnDsyevd	YES	
cusolverDnCheevd	YES	

Function	Migration Support	Diagnostic Message
cusolverDnZheevd	YES	
cusolverDnSsyevdx_bufferSize	YES	
cusolverDnDsyevdx_bufferSize	YES	
cusolverDnCheevdx_bufferSize	YES	
cusolverDnZheevdx_bufferSize	YES	
cusolverDnSsyevdx	YES	
cusolverDnDsyevdx	YES	
cusolverDnCheevdx	YES	
cusolverDnZheevdx	YES	
cusolverDnSsygvd_bufferSize	YES	
cusolverDnDsygvd_bufferSize	YES	
cusolverDnChegvd_bufferSize	YES	
cusolverDnZhegvd_bufferSize	YES	
cusolverDnSsygvd	YES	
cusolverDnDsygvd	YES	
cusolverDnChegvd	YES	
cusolverDnZhegvd	YES	
cusolverDnSsygvdx_bufferSize	YES	
cusolverDnDsygvdx_bufferSize	YES	
cusolverDnChegvdx_bufferSize	YES	
cusolverDnZhegvdx_bufferSize	YES	
cusolverDnSsygvdx	YES	
cusolverDnDsygvdx	YES	
cusolverDnChegvdx	YES	
cusolverDnZhegvdx	YES	
cusolverDnSsyevj_bufferSize	YES	
cusolverDnDsyevj_bufferSize	YES	
cusolverDnCheevj_bufferSize	YES	
cusolverDnZheevj_bufferSize	YES	
cusolverDnSsyevj	YES	

Function	Migration Support	Diagnostic Message
cusolverDnDsyevj	YES	
cusolverDnCheevj	YES	
cusolverDnZheevj	YES	
cusolverDnSsygvj_bufferSize	YES	
cusolverDnDsygvj_bufferSize	YES	
cusolverDnChegvj_bufferSize	YES	
cusolverDnZhegvj_bufferSize	YES	
cusolverDnSsygvj	YES	
cusolverDnDsygvj	YES	
cusolverDnChegvj	YES	
cusolverDnZhegvj	YES	
cusolverDnSsyevjBatched_bufferSize	NO	
cusolverDnDsyevjBatched_bufferSize	NO	
cusolverDnCheevjBatched_bufferSize	NO	
cusolverDnZheevjBatched_bufferSize	NO	
cusolverDnSsyevjBatched	NO	
cusolverDnDsyevjBatched	NO	
cusolverDnCheevjBatched	NO	
cusolverDnZheevjBatched	NO	
cusolverSpCreate	NO	
cusolverSpDestroy	NO	
cusolverSpSetStream	NO	
cusolverSpXcsrissymHost	NO	
cusolverSpScsrlsvluHost	NO	
cusolverSpDcsrlsvluHost	NO	
cusolverSpCcsrlsvluHost	NO	
cusolverSpZcsrlsvluHost	NO	
cusolverSpScsrlsvlu	NO	
cusolverSpDcsrlsvlu	NO	
cusolverSpCcsrlsvlu	NO	

Function	Migration Support	Diagnostic Message
cusolverSpZcsrlsvlu	NO	
cusolverSpScsrlsvqr	NO	
cusolverSpDcsrlsvqr	NO	
cusolverSpCcsrlsvqr	NO	
cusolverSpZcsrlsvqr	NO	
cusolverSpScsrlsvqrHost	NO	
cusolverSpDcsrlsvqrHost	NO	
cusolverSpCcsrlsvqrHost	NO	
cusolverSpZcsrlsvqrHost	NO	
cusolverSpScsrlsvchol	NO	
cusolverSpDcsrlsvchol	NO	
cusolverSpCcsrlsvchol	NO	
cusolverSpZcsrlsvchol	NO	
cusolverSpScsrlsvcholHost	NO	
cusolverSpDcsrlsvcholHost	NO	
cusolverSpCcsrlsvcholHost	NO	
cusolverSpZcsrlsvcholHost	NO	
cusolverSpScsrlsqvqr	NO	
cusolverSpDcsrlsqvqr	NO	
cusolverSpCcsrlsqvqr	NO	
cusolverSpZcsrlsqvqr	NO	
cusolverSpScsrlsqvqrHost	NO	
cusolverSpDcsrlsqvqrHost	NO	
cusolverSpCcsrlsqvqrHost	NO	
cusolverSpZcsrlsqvqrHost	NO	
cusolverSpScsreigvsi	NO	
cusolverSpDcsreigvsi	NO	
cusolverSpCcsreigvsi	NO	
cusolverSpZcsreigvsi	NO	
cusolverSpScsreigvsiHost	NO	

Function	Migration Support	Diagnostic Message
cusolverSpDcsreigvsiHost	NO	
cusolverSpCcsreigvsiHost	NO	
cusolverSpZcsreigvsiHost	NO	
cusolverspScsreigs	NO	
cusolverSpDcsreigs	NO	
cusolverSpCcsreigs	NO	
cusolverSpZcsreigs	NO	
cusolverspScsreigsHost	NO	
cusolverSpDcsreigsHost	NO	
cusolverSpCcsreigsHost	NO	
cusolverSpZcsreigsHost	NO	
cusolverSpXcsrsymrcmHost	NO	
cusolverSpXcsrsymmdqHost	NO	
cusolverSpXcsrsymamdHost	NO	
cusolverSpXcsrmetisndHost	NO	
cusolverSpScsrzfdHost	NO	
cusolverSpDcsrzfdHost	NO	
cusolverSpCcsrzfdHost	NO	
cusolverSpZcsrzfdHost	NO	
cusolverSpXcsrperm_bufferSizeHost	NO	
cusolverSpXcsrpermHost	NO	
cusolverSpCreateCsrqrInfo	NO	
cusolverSpDestroyCsrqrInfo	NO	
cusolverSpXcsrqrAnalysisBatched	NO	
cusolverSpScsrqrBufferInfoBatched	NO	
cusolverSpDcsrqrBufferInfoBatched	NO	
cusolverSpCcsrqrBufferInfoBatched	NO	
cusolverSpZcsrqrBufferInfoBatched	NO	
cusolverSpScsrqrsvBatched	NO	
cusolverSpDcsrqrsvBatched	NO	

Function	Migration Support	Diagnostic Message
cusolverSpCcsrqrsvBatched	NO	
cusolverSpZcsrqrsvBatched	NO	
cusolverRfAccessBundledFactors	NO	
cusolverRfAnalyze	NO	
cusolverRfSetupDevice	NO	
cusolverRfSetupHost	NO	
cusolverRfCreate	NO	
cusolverRfExtractBundledFactorsHost	NO	
cusolverRfExtractSplitFactorsHost	NO	
cusolverRfDestroy	NO	
cusolverRfGetMatrixFormat	NO	
cusolverRfGetNumericProperties	NO	
cusolverRfGetNumericBoostReport	NO	
cusolverRfGetResetValuesFastMode	NO	
cusolverRfGet_Algs	NO	
cusolverRfRefactor	NO	
cusolverRfResetValues	NO	
cusolverRfSetMatrixFormat	NO	
cusolverRfSetNumericProperties	NO	
cusolverRfSetResetValuesFastMode	NO	
cusolverRfSetAlgs	NO	
cusolverRfSolve	NO	
cusolverRfBatchSetupHost	NO	
cusolverRfBatchAnalyze	NO	
cusolverRfBatchResetValues	NO	
cusolverRfBatchRefactor	NO	
cusolverRfBatchSolve	NO	
cusolverRfBatchZeroPivot	NO	
cusolverDnCCgels	NO	
cusolverDnCCgels_bufferSize	NO	

Function	Migration Support	Diagnostic Message
cusolverDnCCgesv	NO	
cusolverDnCCgesv_bufferSize	NO	
cusolverDnCEgels	NO	
cusolverDnCEgels_bufferSize	NO	
cusolverDnCEgesv	NO	
cusolverDnCEgesv_bufferSize	NO	
cusolverDnCKgels	NO	
cusolverDnCKgels_bufferSize	NO	
cusolverDnCKgesv	NO	
cusolverDnCKgesv_bufferSize	NO	
cusolverDnClaswp	NO	
cusolverDnClauum	NO	
cusolverDnClauum_bufferSize	NO	
cusolverDnCreateParams	YES	DPCT1026 / DPCT1027
cusolverDnCsytri	NO	
cusolverDnCsytri_bufferSize	NO	
cusolverDnCYgels	NO	
cusolverDnCYgels_bufferSize	NO	
cusolverDnCYgesv	NO	
cusolverDnCYgesv_bufferSize	NO	
cusolverDnDBgels	NO	
cusolverDnDBgels_bufferSize	NO	
cusolverDnDBgesv	NO	
cusolverDnDBgesv_bufferSize	NO	
cusolverDnDDgels	NO	
cusolverDnDDgels_bufferSize	NO	
cusolverDnDDgesv	NO	
cusolverDnDDgesv_bufferSize	NO	
cusolverDnDestroyParams	YES	DPCT1026 / DPCT1027
cusolverDnDHgels	NO	

Function	Migration Support	Diagnostic Message
cusolverDnDHgels_bufferSize	NO	
cusolverDnDHgesv	NO	
cusolverDnDHgesv_bufferSize	NO	
cusolverDnDlaswp	NO	
cusolverDnDlauum	NO	
cusolverDnDlauum_bufferSize	NO	
cusolverDnDSgels	NO	
cusolverDnDSgels_bufferSize	NO	
cusolverDnDSgesv	NO	
cusolverDnDSgesv_bufferSize	NO	
cusolverDnDsytri	NO	
cusolverDnDsytri_bufferSize	NO	
cusolverDnDXgels	NO	
cusolverDnDXgels_bufferSize	NO	
cusolverDnDXgesv	NO	
cusolverDnDXgesv_bufferSize	NO	
cusolverDnGeqrf	YES	
cusolverDnGeqrf_bufferSize	YES	
cusolverDnGesvd	YES	
cusolverDnGesvd_bufferSize	YES	
cusolverDnGetrf	YES	DPCT1047
cusolverDnGetrf_bufferSize	YES	
cusolverDnGetrs	YES	
cusolverDnIRSInfosCreate	NO	
cusolverDnIRSInfosDestroy	NO	
cusolverDnIRSInfosGetMaxIters	NO	
cusolverDnIRSInfosGetNiters	NO	
cusolverDnIRSInfosGetOuterNiters	NO	
cusolverDnIRSInfosGetResidualHistory	NO	
cusolverDnIRSInfosRequestResidual	NO	

Function	Migration Support	Diagnostic Message
cusolverDnIRSParamsCreate	NO	
cusolverDnIRSParamsDestroy	NO	
cusolverDnIRSParamsDisableFallback	NO	
cusolverDnIRSParamsEnableFallback	NO	
cusolverDnIRSParamsGetMaxIters	NO	
cusolverDnIRSParamsSetMaxIters	NO	
cusolverDnIRSParamsSetMaxItersInner	NO	
cusolverDnIRSParamsSetRefinementSolver	NO	
cusolverDnIRSParamsSetSolverLowestPrecision	NO	
cusolverDnIRSParamsSetSolverMainPrecision	NO	
cusolverDnIRSParamsSetSolverPrecisions	NO	
cusolverDnIRSParamsSetTol	NO	
cusolverDnIRSParamsSetTolInner	NO	
cusolverDnIRSXgels	NO	
cusolverDnIRSXgels_bufferSize	NO	
cusolverDnIRSXgesv	NO	
cusolverDnIRSXgesv_bufferSize	NO	
cusolverDnPotrf	YES	
cusolverDnPotrf_bufferSize	YES	
cusolverDnPotrs	YES	
cusolverDnSBgels	NO	
cusolverDnSBgels_bufferSize	NO	
cusolverDnSBgesv	NO	
cusolverDnSBgesv_bufferSize	NO	
cusolverDnSetAdvOptions	YES	DPCT1026 / DPCT1027
cusolverDnSHgels	NO	
cusolverDnSHgels_bufferSize	NO	
cusolverDnSHgesv	NO	

Function	Migration Support	Diagnostic Message
cusolverDnSHgesv_bufferSize	NO	
cusolverDnSlaswp	NO	
cusolverDnSlauum	NO	
cusolverDnSlauum_bufferSize	NO	
cusolverDnSSgels	NO	
cusolverDnSSgels_bufferSize	NO	
cusolverDnSSgesv	NO	
cusolverDnSSgesv_bufferSize	NO	
cusolverDnSsytri	NO	
cusolverDnSsytri_bufferSize	NO	
cusolverDnSXgels	NO	
cusolverDnSXgels_bufferSize	NO	
cusolverDnSXgesv	NO	
cusolverDnSXgesv_bufferSize	NO	
cusolverDnSyevd	YES	
cusolverDnSyevd_bufferSize	YES	
cusolverDnSyevdx	YES	
cusolverDnSyevdx_bufferSize	YES	
cusolverDnXgeqrf	YES	
cusolverDnXgeqrf_bufferSize	YES	
cusolverDnXgesvd	YES	
cusolverDnXgesvd_bufferSize	YES	
cusolverDnXgesvdp	NO	
cusolverDnXgesvdp_bufferSize	NO	
cusolverDnXgesvdr	NO	
cusolverDnXgesvdr_bufferSize	NO	
cusolverDnXgetrf	YES	DPCT1047
cusolverDnXgetrf_bufferSize	YES	
cusolverDnXgetrs	YES	
cusolverDnXpotrf	YES	

Function	Migration Support	Diagnostic Message
cusolverDnXpotrf_bufferSize	YES	
cusolverDnXpotrs	YES	
cusolverDnXsyevd	YES	
cusolverDnXsyevd_bufferSize	YES	
cusolverDnXsyevdx	YES	
cusolverDnXsyevdx_bufferSize	YES	
cusolverDnXsytrs	NO	
cusolverDnXsytrs_bufferSize	NO	
cusolverDnXtrtri	YES	
cusolverDnXtrtri_bufferSize	YES	
cusolverDnZCgels	NO	
cusolverDnZCgels_bufferSize	NO	
cusolverDnZCgesv	NO	
cusolverDnZCgesv_bufferSize	NO	
cusolverDnZEgels	NO	
cusolverDnZEgels_bufferSize	NO	
cusolverDnZEgesv	NO	
cusolverDnZEgesv_bufferSize	NO	
cusolverDnZKgels	NO	
cusolverDnZKgels_bufferSize	NO	
cusolverDnZKgesv	NO	
cusolverDnZKgesv_bufferSize	NO	
cusolverDnZlaswp	NO	
cusolverDnZlauum	NO	
cusolverDnZlauum_bufferSize	NO	
cusolverDnZsytri	NO	
cusolverDnZsytri_bufferSize	NO	
cusolverDnZYgels	NO	
cusolverDnZYgels_bufferSize	NO	
cusolverDnZYgesv	NO	

Function	Migration Support	Diagnostic Message
cusolverDnZYgesv_bufferSize	NO	
cusolverDnZZgels	NO	
cusolverDnZZgels_bufferSize	NO	
cusolverDnZZgesv	NO	
cusolverDnZZgesv_bufferSize	NO	
cusolverMgCreate	NO	
cusolverMgCreateDeviceGrid	NO	
cusolverMgCreateMatDescr	NO	
cusolverMgDestroy	NO	
cusolverMgDestroyGrid	NO	
cusolverMgDestroyMatrixDesc	NO	
cusolverMgDeviceSelect	NO	
cusolverMgGetrf	NO	
cusolverMgGetrs	NO	
cusolverMgPotrf	NO	
cusolverMgPotri	NO	
cusolverMgPotrs	NO	
cusolverMgSyevd	NO	
cusolverSpScsreigs	NO	
cusolverSpScsreigsHost	NO	
cusolverSpXcsrissym	NO	
cusolverSpXcsrqrBatchedHost	NO	
cusolverSpXcsrzfdHost	NO	
cusolverGetProperty	NO	
cusolverGetVersion	NO	
cusolverDnSetDeterministicMode	NO	
cusolverDnGetDeterministicMode	NO	
cusolverDnXlarft_bufferSize	NO	
cusolverDnXlarft	NO	
cusolverDnLoggerSetCallback	NO	

Function	Migration Support	Diagnostic Message
cusolverDnLoggerSetFile	NO	
cusolverDnLoggerOpenFile	NO	
cusolverDnLoggerSetLevel	NO	
cusolverDnLoggerSetMask	NO	
cusolverDnLoggerForceDisable	NO	
cusolverSpCreateCsrluInfoHost	NO	
cusolverSpDestroyCsrluInfoHost	NO	
cusolverSpXcsrluAnalysisHost	NO	
cusolverSpScsrluBufferInfoHost	NO	
cusolverSpDcsrluBufferInfoHost	NO	
cusolverSpCcsrluBufferInfoHost	NO	
cusolverSpZcsrluBufferInfoHost	NO	
cusolverSpScsrluFactorHost	NO	
cusolverSpDcsrluFactorHost	NO	
cusolverSpCcsrluFactorHost	NO	
cusolverSpZcsrluFactorHost	NO	
cusolverSpScsrluZeroPivotHost	NO	
cusolverSpDcsrluZeroPivotHost	NO	
cusolverSpCcsrluZeroPivotHost	NO	
cusolverSpZcsrluZeroPivotHost	NO	
cusolverSpScsrluSolveHost	NO	
cusolverSpDcsrluSolveHost	NO	
cusolverSpCcsrluSolveHost	NO	
cusolverSpZcsrluSolveHost	NO	
cusolverSpXcsrluNnzHost	NO	
cusolverSpScsrluExtractHost	NO	
cusolverSpDcsrluExtractHost	NO	
cusolverSpCcsrluExtractHost	NO	
cusolverSpZcsrluExtractHost	NO	
cusolverSpCreateCsrqrInfoHost	NO	

Function	Migration Support	Diagnostic Message
cusolverSpDestroyCsrqrInfoHost	NO	
cusolverSpXcsrqrAnalysisHost	NO	
cusolverSpScsrqrBufferInfoHost	NO	
cusolverSpDcsrqrBufferInfoHost	NO	
cusolverSpCcsrqrBufferInfoHost	NO	
cusolverSpZcsrqrBufferInfoHost	NO	
cusolverSpScsrqrSetupHost	NO	
cusolverSpDcsrqrSetupHost	NO	
cusolverSpCcsrqrSetupHost	NO	
cusolverSpZcsrqrSetupHost	NO	
cusolverSpScsrqrFactorHost	NO	
cusolverSpDcsrqrFactorHost	NO	
cusolverSpCcsrqrFactorHost	NO	
cusolverSpZcsrqrFactorHost	NO	
cusolverSpScsrqrZeroPivotHost	NO	
cusolverSpDcsrqrZeroPivotHost	NO	
cusolverSpCcsrqrZeroPivotHost	NO	
cusolverSpZcsrqrZeroPivotHost	NO	
cusolverSpScsrqrSolveHost	NO	
cusolverSpDcsrqrSolveHost	NO	
cusolverSpCcsrqrSolveHost	NO	
cusolverSpZcsrqrSolveHost	NO	
cusolverSpXcsrqrAnalysis	NO	
cusolverSpScsrqrBufferInfo	NO	
cusolverSpDcsrqrBufferInfo	NO	
cusolverSpCcsrqrBufferInfo	NO	
cusolverSpZcsrqrBufferInfo	NO	
cusolverSpScsrqrSetup	NO	
cusolverSpDcsrqrSetup	NO	
cusolverSpCcsrqrSetup	NO	

Function	Migration Support	Diagnostic Message
cusolverSpZcsrqrSetup	NO	
cusolverSpScsrqrFactor	NO	
cusolverSpDcsrqrFactor	NO	
cusolverSpCcsrqrFactor	NO	
cusolverSpZcsrqrFactor	NO	
cusolverSpScsrqrZeroPivot	NO	
cusolverSpDcsrqrZeroPivot	NO	
cusolverSpCcsrqrZeroPivot	NO	
cusolverSpZcsrqrZeroPivot	NO	
cusolverSpScsrqrSolve	NO	
cusolverSpDcsrqrSolve	NO	
cusolverSpCcsrqrSolve	NO	
cusolverSpZcsrqrSolve	NO	
cusolverSpCreateCsrcholInfoHost	NO	
cusolverSpDestroyCsrcholInfoHost	NO	
cusolverSpXcsrcholAnalysisHost	NO	
cusolverSpScsrcholBufferInfoHost	NO	
cusolverSpDcsrcholBufferInfoHost	NO	
cusolverSpCcsrcholBufferInfoHost	NO	
cusolverSpZcsrcholBufferInfoHost	NO	
cusolverSpScsrcholFactorHost	NO	
cusolverSpDcsrcholFactorHost	NO	
cusolverSpCcsrcholFactorHost	NO	
cusolverSpZcsrcholFactorHost	NO	
cusolverSpScsrcholZeroPivotHost	NO	
cusolverSpDcsrcholZeroPivotHost	NO	
cusolverSpCcsrcholZeroPivotHost	NO	
cusolverSpZcsrcholZeroPivotHost	NO	
cusolverSpScsrcholSolveHost	NO	
cusolverSpDcsrcholSolveHost	NO	

Function	Migration Support	Diagnostic Message
cusolverSpCcsrcholSolveHost	NO	
cusolverSpZcsrcholSolveHost	NO	
cusolverSpCreateCsrcholInfo	NO	
cusolverSpDestroyCsrcholInfo	NO	
cusolverSpXcsrcholAnalysis	NO	
cusolverSpScsrcholBufferInfo	NO	
cusolverSpDcsrcholBufferInfo	NO	
cusolverSpCcsrcholBufferInfo	NO	
cusolverSpZcsrcholBufferInfo	NO	
cusolverSpScsrcholFactor	NO	
cusolverSpDcsrcholFactor	NO	
cusolverSpCcsrcholFactor	NO	
cusolverSpZcsrcholFactor	NO	
cusolverSpScsrcholZeroPivot	NO	
cusolverSpDcsrcholZeroPivot	NO	
cusolverSpCcsrcholZeroPivot	NO	
cusolverSpZcsrcholZeroPivot	NO	
cusolverSpScsrcholSolve	NO	
cusolverSpDcsrcholSolve	NO	
cusolverSpCcsrcholSolve	NO	
cusolverSpZcsrcholSolve	NO	
cusolverSpScsrcholDiag	NO	
cusolverSpDcsrcholDiag	NO	
cusolverSpCcsrcholDiag	NO	
cusolverSpZcsrcholDiag	NO	
cusolverMgCreateMatrixDesc	NO	
cusolverMgSyevd_bufferSize	NO	
cusolverMgGetrf_bufferSize	NO	
cusolverMgGetrs_bufferSize	NO	
cusolverMgPotrf_bufferSize	NO	

Function	Migration Support	Diagnostic Message
cusolverMgPotrs_bufferSize	NO	
cusolverMgPotri_bufferSize	NO	
cusolverRfGetAlgs	NO	
cusolverRfAccessBundledFactorsDevice	NO	
cusolverSpGetStream	NO	

cuSPARSE API

Function	Migration Support	Diagnostic Message
cusparseCreate	YES	
cusparseCreateSolveAnalysisInfo	YES	DPCT1026 / DPCT1027
cusparseCreateHybMat	NO	
cusparseCreateMatDescr	YES	
cusparseDestroy	YES	
cusparseDestroySolveAnalysisInfo	YES	DPCT1026 / DPCT1027
cusparseDestroyHybMat	NO	
cusparseDestroyMatDescr	YES	DPCT1026 / DPCT1027
cusparseGetLevelInfo	NO	
cusparseGetMatDiagType	YES	
cusparseGetMatFillMode	YES	
cusparseGetMatIndexBase	YES	
cusparseGetMatType	YES	
cusparseGetPointerMode	YES	DPCT1026 / DPCT1027
cusparseGetVersion	NO	
cusparseSetMatDiagType	YES	
cusparseSetMatFillMode	YES	
cusparseSetMatIndexBase	YES	
cusparseSetMatType	YES	
cusparseSetPointerMode	YES	DPCT1026 / DPCT1027
cusparseSetStream	YES	
cusparseGetStream	YES	

Function	Migration Support	Diagnostic Message
cusparseCreateCsrsv2Info	YES	
cusparseDestroyCsrsv2Info	YES	
cusparseCreateCsrsm2Info	NO	
cusparseDestroyCsrsm2Info	NO	
cusparseCreateCsric02Info	NO	
cusparseDestroyCsric02Info	NO	
cusparseCreateCsrilu02Info	NO	
cusparseDestroyCsrilu02Info	NO	
cusparseCreateBsrsv2Info	NO	
cusparseDestroyBsrsv2Info	NO	
cusparseCreateBsrsm2Info	NO	
cusparseDestroyBsrsm2Info	NO	
cusparseCreateBsric02Info	NO	
cusparseDestroyBsric02Info	NO	
cusparseCreateBsrilu02Info	NO	
cusparseDestroyBsrilu02Info	NO	
cusparseCreateCsrgemm2Info	NO	
cusparseDestroyCsrgemm2Info	NO	
cusparseCreatePruneInfo	NO	
cusparseDestroyPruneInfo	NO	
cusparseSaxpyi	NO	
cusparseDaxpyi	NO	
cusparseCaxpyi	NO	
cusparseZaxpyi	NO	
cusparseSdoti	NO	
cusparseDdoti	NO	
cusparseCdoti	NO	
cusparseZdoti	NO	
cusparseCdotci	NO	
cusparseZdotci	NO	

Function	Migration Support	Diagnostic Message
cusparseSgthr	NO	
cusparseDgthr	NO	
cusparseCgthr	NO	
cusparseZgthr	NO	
cusparseSgthrz	NO	
cusparseDgthrz	NO	
cusparseCgthrz	NO	
cusparseZgthrz	NO	
cusparseSroti	NO	
cusparseDroti	NO	
cusparseSsctr	NO	
cusparseDsctr	NO	
cusparseCsctr	NO	
cusparseZsctr	NO	
cusparseSbsrmv	NO	
cusparseDbsrmv	NO	
cusparseCbsrmv	NO	
cusparseZbsrmv	NO	
cusparseSbsrxmv	NO	
cusparseDbsrxmv	NO	
cusparseCbsrxmv	NO	
cusparseZbsrxmv	NO	
cusparseSbsrsv2_bufferSize	NO	
cusparseDbsrsv2_bufferSize	NO	
cusparseCbsrsv2_bufferSize	NO	
cusparseZbsrsv2_bufferSize	NO	
cusparseSbsrsv2_analysis	NO	
cusparseDbsrsv2_analysis	NO	
cusparseCbsrsv2_analysis	NO	
cusparseZbsrsv2_analysis	NO	

Function	Migration Support	Diagnostic Message
cusparseSbsrsv2_solve	NO	
cusparseDbsrsv2_solve	NO	
cusparseCbsrsv2_solve	NO	
cusparseZbsrsv2_solve	NO	
cusparseXbsrsv2_zeroPivot	NO	
cusparseScsrmv	YES	DPCT1045
cusparseDcsrmv	YES	DPCT1045
cusparseCcsrmv	YES	DPCT1045
cusparseZcsrmv	YES	DPCT1045
cusparseScsrmv_mp	YES	DPCT1045
cusparseDcsrmv_mp	YES	DPCT1045
cusparseCcsrmv_mp	YES	DPCT1045
cusparseZcsrmv_mp	YES	DPCT1045
cusparseCsrmvEx	YES	DPCT1045
cusparseCsrmvEx_bufferSize	YES	DPCT1045
cusparseScsrsv_analysis	YES	DPCT1026 / DPCT1027
cusparseDcsrsv_analysis	YES	DPCT1026 / DPCT1027
cusparseCcsrsv_analysis	YES	DPCT1026 / DPCT1027
cusparseZcsrsv_analysis	YES	DPCT1026 / DPCT1027
cusparseCsrsv_analysisEx	YES	
cusparseScsrsv_solve	YES	
cusparseDcsrsv_solve	YES	
cusparseCcsrsv_solve	YES	
cusparseZcsrsv_solve	YES	
cusparseCsrsv_solveEx	YES	
cusparseScsrsv2_bufferSize	YES	
cusparseDcsrsv2_bufferSize	YES	
cusparseCcsrsv2_bufferSize	YES	
cusparseZcsrsv2_bufferSize	YES	
cusparseScsrsv2_analysis	YES	

Function	Migration Support	Diagnostic Message
cusparseDcsrsv2_analysis	YES	
cusparseCcsrsv2_analysis	YES	
cusparseZcsrsv2_analysis	YES	
cusparseScsrsv2_solve	YES	
cusparseDcsrsv2_solve	YES	
cusparseCcsrsv2_solve	YES	
cusparseZcsrsv2_solve	YES	
cusparseXcsrsv2_zeroPivot	NO	
cusparseSgemvi	NO	
cusparseDgemvi	NO	
cusparseCgemvi	NO	
cusparseZgemvi	NO	
cusparseSgemvi_bufferSize	NO	
cusparseDgemvi_bufferSize	NO	
cusparseCgemvi_bufferSize	NO	
cusparseZgemvi_bufferSize	NO	
cusparseShybmv	NO	
cusparseDhybmv	NO	
cusparseChybmv	NO	
cusparseZhybmv	NO	
cusparseShybsv_analysis	NO	
cusparseDhybsv_analysis	NO	
cusparseChybsv_analysis	NO	
cusparseZhybsv_analysis	NO	
cusparseShybsv_solve	NO	
cusparseDhybsv_solve	NO	
cusparseChybsv_solve	NO	
cusparseZhybsv_solve	NO	
cusparseSbsrmm	NO	
cusparseDbsrmm	NO	

Function	Migration Support	Diagnostic Message
cusparseCbsrmm	NO	
cusparseZbsrmm	NO	
cusparseSbsrsm2_bufferSize	NO	
cusparseDbsrsm2_bufferSize	NO	
cusparseCbsrsm2_bufferSize	NO	
cusparseZbsrsm2_bufferSize	NO	
cusparseSbsrsm2_analysis	NO	
cusparseDbsrsm2_analysis	NO	
cusparseCbsrsm2_analysis	NO	
cusparseZbsrsm2_analysis	NO	
cusparseSbsrsm2_solve	NO	
cusparseDbsrsm2_solve	NO	
cusparseCbsrsm2_solve	NO	
cusparseZbsrsm2_solve	NO	
cusparseXbsrsm2_zeroPivot	NO	
cusparseScsrmm	YES	DPCT1045
cusparseDcsrmm	YES	DPCT1045
cusparseCcsrmm	YES	DPCT1045
cusparseZcsrmm	YES	DPCT1045
cusparseScsrmm2	YES	DPCT1045
cusparseDcsrmm2	YES	DPCT1045
cusparseCcsrmm2	YES	DPCT1045
cusparseZcsrmm2	YES	DPCT1045
cusparseScsrsm_analysis	NO	
cusparseDcsrsm_analysis	NO	
cusparseCcsrsm_analysis	NO	
cusparseZcsrsm_analysis	NO	
cusparseScsrsm_solve	NO	
cusparseDcsrsm_solve	NO	
cusparseCcsrsm_solve	NO	

Function	Migration Support	Diagnostic Message
cusparseZcsrsm_solve	NO	
cusparseScsrsm2_bufferSizeExt	NO	
cusparseDcsrsm2_bufferSizeExt	NO	
cusparseCcsrsm2_bufferSizeExt	NO	
cusparseZcsrsm2_bufferSizeExt	NO	
cusparseScsrsm2_analysis	NO	
cusparseDcsrsm2_analysis	NO	
cusparseCcsrsm2_analysis	NO	
cusparseZcsrsm2_analysis	NO	
cusparseScsrsm2_solve	NO	
cusparseDcsrsm2_solve	NO	
cusparseCcsrsm2_solve	NO	
cusparseZcsrsm2_solve	NO	
cusparseXcsrsm2_zeroPivot	NO	
cusparseSgemmi	NO	
cusparseDgemmi	NO	
cusparseCgemmi	NO	
cusparseZgemmi	NO	
cusparseXcsrgeamNnz	NO	
cusparseScsrgeam	NO	
cusparseDcsrgeam	NO	
cusparseCcsrgeam	NO	
cusparseZcsrgeam	NO	
cusparseScsrgeam2_bufferSizeExt	NO	
cusparseDcsrgeam2_bufferSizeExt	NO	
cusparseCcsrgeam2_bufferSizeExt	NO	
cusparseZcsrgeam2_bufferSizeExt	NO	
cusparseXcsrgeam2Nnz	NO	
cusparseScsrgeam2	NO	
cusparseDcsrgeam2	NO	

Function	Migration Support	Diagnostic Message
cusparseCcsrgeam2	NO	
cusparseZcsrgeam2	NO	
cusparseXcsrgemmNnz	NO	
cusparseScsrgemm	NO	
cusparseDcsrgemm	NO	
cusparseCcsrgemm	NO	
cusparseZcsrgemm	NO	
cusparseScsrgemm2_bufferSizeExt	NO	
cusparseDcsrgemm2_bufferSizeExt	NO	
cusparseCcsrgemm2_bufferSizeExt	NO	
cusparseZcsrgemm2_bufferSizeExt	NO	
cusparseXcsrgemm2Nnz	NO	
cusparseScsrgemm2	NO	
cusparseDcsrgemm2	NO	
cusparseCcsrgemm2	NO	
cusparseZcsrgemm2	NO	
cusparseScsric0	NO	
cusparseDcsric0	NO	
cusparseCcsric0	NO	
cusparseZcsric0	NO	
cusparseScsric02_bufferSize	NO	
cusparseDcsric02_bufferSize	NO	
cusparseCcsric02_bufferSize	NO	
cusparseZcsric02_bufferSize	NO	
cusparseScsric02_analysis	NO	
cusparseDcsric02_analysis	NO	
cusparseCcsric02_analysis	NO	
cusparseZcsric02_analysis	NO	
cusparseScsric02	NO	
cusparseDcsric02	NO	

Function	Migration Support	Diagnostic Message
cusparseCcsric02	NO	
cusparseZcsric02	NO	
cusparseXcsric02_zeroPivot	NO	
cusparseSbsric02_bufferSize	NO	
cusparseDbsric02_bufferSize	NO	
cusparseCbsric02_bufferSize	NO	
cusparseZbsric02_bufferSize	NO	
cusparseSbsric02_analysis	NO	
cusparseDbsric02_analysis	NO	
cusparseCbsric02_analysis	NO	
cusparseZbsric02_analysis	NO	
cusparseSbsric02	NO	
cusparseDbsric02	NO	
cusparseCbsric02	NO	
cusparseZbsric02	NO	
cusparseXbsric02_zeroPivot	NO	
cusparseScsrilu0	NO	
cusparseDcsrilu0	NO	
cusparseCcsrilu0	NO	
cusparseZcsrilu0	NO	
cusparseCsrilu0Ex	NO	
cusparseScsrilu02_numericBoost	NO	
cusparseDcsrilu02_numericBoost	NO	
cusparseCcsrilu02_numericBoost	NO	
cusparseZcsrilu02_numericBoost	NO	
cusparseScsrilu02_bufferSize	NO	
cusparseDcsrilu02_bufferSize	NO	
cusparseCcsrilu02_bufferSize	NO	
cusparseZcsrilu02_bufferSize	NO	
cusparseScsrilu02_analysis	NO	

Function	Migration Support	Diagnostic Message
cusparseDcsrilu02_analysis	NO	
cusparseCcsrilu02_analysis	NO	
cusparseZcsrilu02_analysis	NO	
cusparseScsrilu02	NO	
cusparseDcsrilu02	NO	
cusparseCcsrilu02	NO	
cusparseZcsrilu02	NO	
cusparseXcsrilu02_zeroPivot	NO	
cusparseSbsrilu02_numericBoost	NO	
cusparseDbsrilu02_numericBoost	NO	
cusparseCbsrilu02_numericBoost	NO	
cusparseZbsrilu02_numericBoost	NO	
cusparseSbsrilu02_bufferSize	NO	
cusparseDbsrilu02_bufferSize	NO	
cusparseCbsrilu02_bufferSize	NO	
cusparseZbsrilu02_bufferSize	NO	
cusparseSbsrilu02_analysis	NO	
cusparseDbsrilu02_analysis	NO	
cusparseCbsrilu02_analysis	NO	
cusparseZbsrilu02_analysis	NO	
cusparseSbsrilu02	NO	
cusparseDbsrilu02	NO	
cusparseCbsrilu02	NO	
cusparseZbsrilu02	NO	
cusparseXbsrilu02_zeroPivot	NO	
cusparseSgtsv	NO	
cusparseDgtsv	NO	
cusparseCgtsv	NO	
cusparseZgtsv	NO	
cusparseSgtsv_nopivot	NO	

Function	Migration Support	Diagnostic Message
cusparseDgtsv_nopivot	NO	
cusparseCgtsv_nopivot	NO	
cusparseZgtsv_nopivot	NO	
cusparseSgtsv2_bufferSizeExt	NO	
cusparseDgtsv2_bufferSizeExt	NO	
cusparseCgtsv2_bufferSizeExt	NO	
cusparseZgtsv2_bufferSizeExt	NO	
cusparseSgtsv2	NO	
cusparseDgtsv2	NO	
cusparseCgtsv2	NO	
cusparseZgtsv2	NO	
cusparseSgtsv2_nopivot_bufferSizeExt	NO	
cusparseDgtsv2_nopivot_bufferSizeExt	NO	
cusparseCgtsv2_nopivot_bufferSizeExt	NO	
cusparseZgtsv2_nopivot_bufferSizeExt	NO	
cusparseSgtsv2_nopivot	NO	
cusparseDgtsv2_nopivot	NO	
cusparseCgtsv2_nopivot	NO	
cusparseZgtsv2_nopivot	NO	
cusparseSgtsvStridedBatch	NO	
cusparseDgtsvStridedBatch	NO	
cusparseCgtsvStridedBatch	NO	
cusparseZgtsvStridedBatch	NO	
cusparseSgtsv2StridedBatch_bufferSizeExt	NO	
cusparseDgtsv2StridedBatch_bufferSizeExt	NO	
cusparseCgtsv2StridedBatch_bufferSizeExt	NO	
cusparseZgtsv2StridedBatch_bufferSizeExt	NO	
cusparseSgtsv2StridedBatch	NO	

Function	Migration Support	Diagnostic Message
cusparseDgtsv2StridedBatch	NO	
cusparseCgtsv2StridedBatch	NO	
cusparseZgtsv2StridedBatch	NO	
cusparseSgtsvInterleavedBatch_bufferSizeExt	NO	
cusparseDgtsvInterleavedBatch_bufferSizeExt	NO	
cusparseCgtsvInterleavedBatch_bufferSizeExt	NO	
cusparseZgtsvInterleavedBatch_bufferSizeExt	NO	
cusparseSgtsvInterleavedBatch	NO	
cusparseDgtsvInterleavedBatch	NO	
cusparseCgtsvInterleavedBatch	NO	
cusparseZgtsvInterleavedBatch	NO	
cusparseSgpsvInterleavedBatch_bufferSizeExt	NO	
cusparseDgpsvInterleavedBatch_bufferSizeExt	NO	
cusparseCgpsvInterleavedBatch_bufferSizeExt	NO	
cusparseZgpsvInterleavedBatch_bufferSizeExt	NO	
cusparseSgpsvInterleavedBatch	NO	
cusparseDgpsvInterleavedBatch	NO	
cusparseCgpsvInterleavedBatch	NO	
cusparseZgpsvInterleavedBatch	NO	
cusparseScsrcolor	NO	
cusparseDcsrcolor	NO	
cusparseCcsrcolor	NO	
cusparseZcsrcolor	NO	
cusparseSbsr2csr	NO	
cusparseDbsr2csr	NO	
cusparseCbsr2csr	NO	

Function	Migration Support	Diagnostic Message
cusparseZbsr2csr	NO	
cusparseSgebsr2gebsc_bufferSize	NO	
cusparseDgebsr2gebsc_bufferSize	NO	
cusparseCgebsr2gebsc_bufferSize	NO	
cusparseZgebsr2gebsc_bufferSize	NO	
cusparseSgebsr2gebsc	NO	
cusparseDgebsr2gebsc	NO	
cusparseCgebsr2gebsc	NO	
cusparseZgebsr2gebsc	NO	
cusparseSgebsr2gebsr_bufferSize	NO	
cusparseDgebsr2gebsr_bufferSize	NO	
cusparseCgebsr2gebsr_bufferSize	NO	
cusparseZgebsr2gebsr_bufferSize	NO	
cusparseXgebsr2gebsrNnz	NO	
cusparseSgebsr2gebsr	NO	
cusparseDgebsr2gebsr	NO	
cusparseCgebsr2gebsr	NO	
cusparseZgebsr2gebsr	NO	
cusparseSgebsr2csr	NO	
cusparseDgebsr2csr	NO	
cusparseCgebsr2csr	NO	
cusparseZgebsr2csr	NO	
cusparseScsr2gebsr_bufferSize	NO	
cusparseDcsr2gebsr_bufferSize	NO	
cusparseCcsr2gebsr_bufferSize	NO	
cusparseZcsr2gebsr_bufferSize	NO	
cusparseXcsr2gebsrNnz	NO	
cusparseScsr2gebsr	NO	
cusparseDcsr2gebsr	NO	
cusparseCcsr2gebsr	NO	

Function	Migration Support	Diagnostic Message
cusparseZcsr2gebsr	NO	
cusparseXcoo2csr	NO	
cusparseScsc2dense	NO	
cusparseDcsc2dense	NO	
cusparseCcsc2dense	NO	
cusparseZcsc2dense	NO	
cusparseScsc2hyb	NO	
cusparseDcsc2hyb	NO	
cusparseCcsc2hyb	NO	
cusparseZcsc2hyb	NO	
cusparseXcsr2bsrNnz	NO	
cusparseScsr2bsr	NO	
cusparseDcsr2bsr	NO	
cusparseCcsr2bsr	NO	
cusparseZcsr2bsr	NO	
cusparseXcsr2coo	NO	
cusparseScsr2csc	YES	
cusparseDcsr2csc	YES	
cusparseCcsr2csc	YES	
cusparseZcsr2csc	YES	
cusparseCsr2cscEx	NO	
cusparseCsr2cscEx2_bufferSize	YES	
cusparseCsr2cscEx2	YES	
cusparseScsr2dense	NO	
cusparseDcsr2dense	NO	
cusparseCcsr2dense	NO	
cusparseZcsr2dense	NO	
cusparseScsr2csr_compress	NO	
cusparseDcsr2csr_compress	NO	
cusparseCcsr2csr_compress	NO	

Function	Migration Support	Diagnostic Message
cusparseZcsr2csr_compress	NO	
cusparseScsr2hyb	NO	
cusparseDcsr2hyb	NO	
cusparseCcsr2hyb	NO	
cusparseZcsr2hyb	NO	
cusparseSdense2csc	NO	
cusparseDdense2csc	NO	
cusparseCdense2csc	NO	
cusparseZdense2csc	NO	
cusparseSdense2csr	NO	
cusparseDdense2csr	NO	
cusparseCdense2csr	NO	
cusparseZdense2csr	NO	
cusparseSdense2hyb	NO	
cusparseDdense2hyb	NO	
cusparseCdense2hyb	NO	
cusparseZdense2hyb	NO	
cusparseShyb2csc	NO	
cusparseDhyb2csc	NO	
cusparseChyb2csc	NO	
cusparseZhyb2csc	NO	
cusparseShyb2csr	NO	
cusparseDhyb2csr	NO	
cusparseChyb2csr	NO	
cusparseZhyb2csr	NO	
cusparseShyb2dense	NO	
cusparseDhyb2dense	NO	
cusparseChyb2dense	NO	
cusparseZhyb2dense	NO	
cusparseSnnz	NO	

Function	Migration Support	Diagnostic Message
cusparseDnnz	NO	
cusparseCnnz	NO	
cusparseZnnz	NO	
cusparseCreateIdentityPermutation	NO	
cusparseXcoosort_bufferSizeExt	NO	
cusparseXcoosortByRow	NO	
cusparseXcoosortByColumn	NO	
cusparseXcsrsort_bufferSizeExt	NO	
cusparseXcsrsort	NO	
cusparseXcscsort_bufferSizeExt	NO	
cusparseXcscsort	NO	
cusparseCreateCsru2csrInfo	NO	
cusparseDestroyCsru2csrInfo	NO	
cusparseScsru2csr_bufferSizeExt	NO	
cusparseDcsru2csr_bufferSizeExt	NO	
cusparseCcsru2csr_bufferSizeExt	NO	
cusparseZcsru2csr_bufferSizeExt	NO	
cusparseScsru2csr	NO	
cusparseDcsru2csr	NO	
cusparseCcsru2csr	NO	
cusparseZcsru2csr	NO	
cusparseScsr2csru	NO	
cusparseDcsr2csru	NO	
cusparseCcsr2csru	NO	
cusparseZcsr2csru	NO	
cusparseHpruneDense2csr_bufferSizeExt	NO	
cusparseSpruneDense2csr_bufferSizeExt	NO	
cusparseDpruneDense2csr_bufferSizeExt	NO	
cusparseHpruneDense2csrNnz	NO	
cusparseSpruneDense2csrNnz	NO	

Function	Migration Support	Diagnostic Message
cusparseDpruneDense2csrNnz	NO	
cusparseHpruneDense2csr	NO	
cusparseSpruneDense2csr	NO	
cusparseDpruneDense2csr	NO	
cusparseHpruneCsr2csr_bufferSizeExt	NO	
cusparseSpruneCsr2csr_bufferSizeExt	NO	
cusparseDpruneCsr2csr_bufferSizeExt	NO	
cusparseHpruneCsr2csrNnz	NO	
cusparseSpruneCsr2csrNnz	NO	
cusparseDpruneCsr2csrNnz	NO	
cusparseHpruneCsr2csr	NO	
cusparseSpruneCsr2csr	NO	
cusparseDpruneCsr2csr	NO	
cusparseHpruneDense2csrByPercentage_bufferSizeExt	NO	
cusparseSpruneDense2csrByPercentage_bufferSizeExt	NO	
cusparseDpruneDense2csrByPercentage_bufferSizeExt	NO	
cusparseHpruneDense2csrNnzByPercentage	NO	
cusparseSpruneDense2csrNnzByPercentage	NO	
cusparseDpruneDense2csrNnzByPercentage	NO	
cusparseHpruneDense2csrByPercentage	NO	
cusparseSpruneDense2csrByPercentage	NO	
cusparseDpruneDense2csrByPercentage	NO	
cusparseHpruneCsr2csrByPercentage_bufferSizeExt	NO	
cusparseSpruneCsr2csrByPercentage_bufferSizeExt	NO	
cusparseDpruneCsr2csrByPercentage_bufferSizeExt	NO	

Function	Migration Support	Diagnostic Message
cusparseHpruneCsr2csrNnzByPercentage	NO	
cusparseSpruneCsr2csrNnzByPercentage	NO	
cusparseDpruneCsr2csrNnzByPercentage	NO	
cusparseHpruneCsr2csrByPercentage	NO	
cusparseSpruneCsr2csrByPercentage	NO	
cusparseDpruneCsr2csrByPercentage	NO	
cusparseSnnz_compress	NO	
cusparseDnnz_compress	NO	
cusparseCnnz_compress	NO	
cusparseZnnz_compress	NO	
cusparseCreateSpVec	NO	
cusparseDestroySpVec	NO	
cusparseSpVecGet	NO	
cusparseSpVecGetIndexBase	NO	
cusparseSpVecGetValues	NO	
cusparseSpVecSetValues	NO	
cusparseCreateCoo	YES	
cusparseCreateCooAoS	NO	
cusparseCreateCsr	YES	
cusparseDestroySpMat	YES	
cusparseCooGet	NO	
cusparseCooAoSGet	NO	
cusparseCsrGet	YES	
cusparseSpMatGetFormat	YES	
cusparseSpMatGetIndexBase	YES	
cusparseSpMatGetValues	YES	
cusparseSpMatSetValues	YES	
cusparseSpMatGetStridedBatch	NO	
cusparseSpMatSetStridedBatch	NO	
cusparseCreateDnVec	YES	

Function	Migration Support	Diagnostic Message
cusparseDestroyDnVec	YES	
cusparseDnVecGet	YES	
cusparseDnVecGetValues	YES	
cusparseDnVecSetValues	YES	
cusparseCreateDnMat	YES	
cusparseDestroyDnMat	YES	
cusparseDnMatGet	YES	
cusparseDnMatGetValues	YES	
cusparseDnMatSetValues	YES	
cusparseDnMatGetStridedBatch	NO	
cusparseDnMatSetStridedBatch	NO	
cusparseSpVV_bufferSize	NO	
cusparseSpVV	NO	
cusparseSpMV_bufferSize	YES	
cusparseSpMV	YES	
cusparseSpMM_bufferSize	YES	
cusparseSpMM	YES	
cusparseConstrainedGeMM	NO	
cusparseConstrainedGeMM_bufferSize	NO	
cusparseAxpby	NO	
cusparseBlockedEllGet	NO	
cusparseCbsric02_bufferSizeExt	NO	
cusparseCbsrilu02_bufferSizeExt	NO	
cusparseCbsrsm2_bufferSizeExt	NO	
cusparseCbsrsv2_bufferSizeExt	NO	
cusparseCcsr2gebsr_bufferSizeExt	NO	
cusparseCcsric02_bufferSizeExt	NO	
cusparseCcsrilu02_bufferSizeExt	NO	
cusparseCcsrsv2_bufferSizeExt	YES	
cusparseCgebsr2gebsc_bufferSizeExt	NO	

Function	Migration Support	Diagnostic Message
cusparseCgebsr2gebsr_bufferSizeExt	NO	
cusparseCooSetPointers	NO	
cusparseCooSetStridedBatch	NO	
cusparseCreateBlockedEll	NO	
cusparseCreateColorInfo	NO	
cusparseCreateCsc	NO	
cusparseCscGet	NO	
cusparseCscSetPointers	NO	
cusparseCsrSetPointers	YES	
cusparseCsrSetStridedBatch	NO	
cusparseDbsric02_bufferSizeExt	NO	
cusparseDbsrilu02_bufferSizeExt	NO	
cusparseDbsrsm2_bufferSizeExt	NO	
cusparseDbsrsv2_bufferSizeExt	NO	
cusparseDcsr2gebsr_bufferSizeExt	NO	
cusparseDcsric02_bufferSizeExt	NO	
cusparseDcsrilu02_bufferSizeExt	NO	
cusparseDcsrsv2_bufferSizeExt	YES	
cusparseDenseToSparse_analysis	NO	
cusparseDenseToSparse_bufferSize	NO	
cusparseDenseToSparse_convert	NO	
cusparseDestroyColorInfo	NO	
cusparseDgebsr2gebsc_bufferSizeExt	NO	
cusparseDgebsr2gebsr_bufferSizeExt	NO	
cusparseGather	NO	
cusparseGetErrorName	YES	DPCT1009
cusparseGetErrorString	YES	DPCT1009
cusparseGetProperty	YES	
cusparseLoggerForceDisable	NO	
cusparseLoggerOpenFile	NO	

Function	Migration Support	Diagnostic Message
cusparseLoggerSetCallback	NO	
cusparseLoggerSetFile	NO	
cusparseLoggerSetLevel	NO	
cusparseLoggerSetMask	NO	
cusparseRot	NO	
cusparseSbsric02_bufferSizeExt	NO	
cusparseSbsrilu02_bufferSizeExt	NO	
cusparseSbsrsm2_bufferSizeExt	NO	
cusparseSbsrsv2_bufferSizeExt	NO	
cusparseScatter	NO	
cusparseScsr2gebsr_bufferSizeExt	NO	
cusparseScsric02_bufferSizeExt	NO	
cusparseScsrilu02_bufferSizeExt	NO	
cusparseScsrsv2_bufferSizeExt	YES	
cusparseSDDMM	NO	
cusparseSDDMM_bufferSize	NO	
cusparseSDDMM_preprocess	NO	
cusparseSgebsr2gebsc_bufferSizeExt	NO	
cusparseSgebsr2gebsr_bufferSizeExt	NO	
cusparseSparseToDense	NO	
cusparseSparseToDense_bufferSize	NO	
cusparseSpGEMM_compute	YES	
cusparseSpGEMM_copy	YES	
cusparseSpGEMM_createDescr	YES	
cusparseSpGEMM_destroyDescr	YES	
cusparseSpGEMMreuse_compute	NO	
cusparseSpGEMMreuse_copy	NO	
cusparseSpGEMMreuse_nnz	NO	
cusparseSpGEMMreuse_workEstimation	NO	
cusparseSpGEMM_workEstimation	YES	

Function	Migration Support	Diagnostic Message
cusparseSpMatGetAttribute	YES	
cusparseSpMatGetNumBatches	NO	
cusparseSpMatGetSize	YES	
cusparseSpMatSetAttribute	YES	
cusparseSpMatSetNumBatches	NO	
cusparseSpMMOp	NO	
cusparseSpMMOp_createPlan	NO	
cusparseSpMMOp_destroyPlan	NO	
cusparseSpMM_preprocess	YES	
cusparseSpSM_analysis	YES	
cusparseSpSM_bufferSize	YES	
cusparseSpSM_createDescr	YES	DPCT1026 / DPCT1027
cusparseSpSM_destroyDescr	YES	DPCT1026 / DPCT1027
cusparseSpSM_solve	YES	
cusparseSpSV_analysis	YES	
cusparseSpSV_bufferSize	YES	
cusparseSpSV_createDescr	YES	
cusparseSpSV_destroyDescr	YES	
cusparseSpSV_solve	YES	
cusparseXgebsr2csr	NO	
cusparseZbsric02_bufferSizeExt	NO	
cusparseZbsrilu02_bufferSizeExt	NO	
cusparseZbsrsm2_bufferSizeExt	NO	
cusparseZbsrsv2_bufferSizeExt	NO	
cusparseZcsr2gebsr_bufferSizeExt	NO	
cusparseZcsric02_bufferSizeExt	NO	
cusparseZcsrilu02_bufferSizeExt	NO	
cusparseZcsrsv2_bufferSizeExt	YES	
cusparseZgebsr2gebsc_bufferSizeExt	NO	
cusparseZgebsr2gebsr_bufferSizeExt	NO	

Function	Migration Support	Diagnostic Message
cusparseCreateConstSpVec	NO	
cusparseConstSpVecGetValues	NO	
cusparseCreateConstDnVec	YES	
cusparseConstDnVecGet	YES	
cusparseConstDnVecGetValues	YES	
cusparseConstSpMatGetValues	NO	
cusparseCreateConstCsr	NO	
cusparseCreateConstCsc	NO	
cusparseConstCsrGet	NO	
cusparseConstCscGet	NO	
cusparseCreateConstCoo	NO	
cusparseConstCooGet	NO	
cusparseCreateConstBlockedEll	NO	
cusparseCreateConstDnMat	NO	
cusparseConstDnMatGet	NO	
cusparseConstDnMatGetValues	NO	
cusparseSpGEMM_getNumProducts	NO	
cusparseSpGEMM_estimateMemory	NO	
cusparseConstSpVecGet	NO	
cusparseBsrSetStridedBatch	NO	
cusparseCreateBsr	NO	
cusparseCreateConstBsr	NO	
cusparseConstBlockedEllGet	NO	
cusparseCreateSlicedEll	NO	
cusparseCreateConstSlicedEll	NO	
cusparseSpMV_preprocess	NO	
cusparseSpSV_updateMatrix	NO	
cusparseSpSM_updateMatrix	NO	

NCCL API

Function	Migration Support	Diagnostic Message
ncclGetVersion	YES	
ncclGetUniqueId	YES	
ncclCommInitRank	YES	
ncclCommInitAll	NO	
ncclCommDestroy	YES	
ncclCommAbort	NO	
ncclCommGetAsyncError	YES	DPCT1026 / DPCT1027
ncclCommCount	YES	
ncclCommCuDevice	YES	
ncclCommUserRank	YES	
ncclCommSplit	NO	
ncclCommFinalize	NO	
ncclCommRegister	NO	
ncclCommDeregister	NO	
ncclMemAlloc	NO	
ncclMemFree	NO	
ncclAllReduce	YES	DPCT1067
ncclBroadcast	YES	
ncclReduce	YES	DPCT1067
ncclAllGather	NO	
ncclReduceScatter	YES	DPCT1067
ncclGroupStart	NO	
ncclGroupEnd	NO	
ncclGroupSimulateEnd	NO	
ncclSend	YES	
ncclRecv	YES	
ncclGetLastError	YES	DPCT1026 / DPCT1027
ncclRedOpCreatePreMulSum	NO	
ncclRedOpDestroy	NO	

Function	Migration Support	Diagnostic Message
ncclBcast	YES	
ncclGetErrorString	YES	DPCT1009

nvGRAPH API

Function	Migration Support	Diagnostic Message
nvgraphGetProperty	NO	
nvgraphCreate	NO	
nvgraphDestroy	NO	
nvgraphCreateGraphDescr	NO	
nvgraphDestroyGraphDescr	NO	
nvgraphSetGraphStructure	NO	
nvgraphGetGraphStructure	NO	
nvgraphConvertTopology	NO	
nvgraphConvertGraph	NO	
nvgraphAllocateEdgeData	NO	
nvgraphSetEdgeData	NO	
nvgraphGetEdgeData	NO	
nvgraphAllocateVertexData	NO	
nvgraphSetVertexData	NO	
nvgraphGetVertexData	NO	
nvgraphExtractSubgraphByVertex	NO	
nvgraphExtractSubgraphByEdge	NO	
nvgraphWidestPath	NO	
nvgraphSssp	NO	
nvgraphSrSpmv	NO	
nvgraphPagerank	NO	
nvgraphTriangleCount	NO	
nvgraphStatusGetString	NO	
nvgraphSpectralClustering	NO	
nvgraphAnalyzeClustering	NO	

Function	Migration Support	Diagnostic Message
nvgraphTraversalParameterInit	NO	
nvgraphTraversalSetDistancesIndex	NO	
nvgraphTraversalGetDistancesIndex	NO	
nvgraphTraversalSetPredecessorsIndex	NO	
nvgraphTraversalGetPredecessorsIndex	NO	
nvgraphTraversalSetEdgeMaskIndex	NO	
nvgraphTraversalGetEdgeMaskIndex	NO	
nvgraphTraversalSetUndirectedFlag	NO	
nvgraphTraversalGetUndirectedFlag	NO	
nvgraphTraversalSetAlpha	NO	
nvgraphTraversalGetAlpha	NO	
nvgraphTraversalSetBeta	NO	
nvgraphTraversalGetBeta	NO	
nvgraphTraversal	NO	

nvJPEG API

Function	Migration Support	Diagnostic Message
nvjpegGetProperty	NO	
nvjpegCreate	NO	
nvjpegCreateSimple	NO	
nvjpegCreateEx	NO	
nvjpegDestroy	NO	
nvjpegSetDeviceMemoryPadding	NO	
nvjpegGetDeviceMemoryPadding	NO	
nvjpegSetPinnedMemoryPadding	NO	
nvjpegGetPinnedMemoryPadding	NO	
nvjpegJpegStateCreate	NO	
nvjpegJpegStateDestroy	NO	
nvjpegGetImageInfo	NO	
nvjpegDecode	NO	

Function	Migration Support	Diagnostic Message
nvjpegDecodeBatchedInitialize	NO	
nvjpegDecodeBatched	NO	
nvjpegDecodePhaseOne	NO	
nvjpegDecodePhaseTwo	NO	
nvjpegDecodePhaseThree	NO	
nvjpegDecodeBatchedPhaseOne	NO	
nvjpegDecodeBatchedPhaseTwo	NO	
nvjpegDecodeBatchedPhaseThree	NO	

NVML API

Function	Migration Support	Diagnostic Message
nvmlDeviceGetAccountingMode	NO	
nvmlDeviceGetAccountingStats	NO	
nvmlDeviceGetAccountingPids	NO	
nvmlDeviceGetAccountingBufferSize	NO	
nvmlDeviceSetAccountingMode	NO	
nvmlDeviceClearAccountingPids	NO	
nvmlInit	NO	
nvmlInitWithFlags	NO	
nvmlShutdown	NO	
nvmlErrorString	NO	
nvmlSystemGetDriverVersion	NO	
nvmlSystemGetNVMLVersion	NO	
nvmlSystemGetCudaDriverVersion	NO	
nvmlSystemGetCudaDriverVersion	NO	
nvmlSystemGetProcessName	NO	
nvmlUnitGetCount	NO	
nvmlUnitGetHandleByIndex	NO	
nvmlUnitGetUnitInfo	NO	
nvmlUnitGetLedState	NO	

Function	Migration Support	Diagnostic Message
nvmlUnitGetPsuInfo	NO	
nvmlUnitGetTemperature	NO	
nvmlUnitGetFanSpeedInfo	NO	
nvmlUnitGetDevices	NO	
nvmlSystemGetHicVersion	NO	
nvmlDeviceGetCount	NO	
nvmlDeviceGetAttributes	NO	
nvmlDeviceGetHandleByIndex	NO	
nvmlDeviceGetHandleBySerial	NO	
nvmlDeviceGetHandleByUUID	NO	
nvmlDeviceGetHandleByPciBusId	NO	
nvmlDeviceGetName	NO	
nvmlDeviceGetBrand	NO	
nvmlDeviceGetIndex	NO	
nvmlDeviceGetSerial	NO	
nvmlDeviceGetTopologyCommonAncestor	NO	
nvmlDeviceGetTopologyNearestGpus	NO	
nvmlSystemGetTopologyGpuSet	NO	
nvmlDeviceGetP2PStatus	NO	
nvmlDeviceGetUUID	NO	
nvmlVgpuInstanceGetMdevUUID	NO	
nvmlDeviceGetMinorNumber	NO	
nvmlDeviceGetBoardPartNumber	NO	
nvmlDeviceGetInforomVersion	NO	
nvmlDeviceGetInforomImageVersion	NO	
nvmlDeviceGetInforomConfigurationChecksum	NO	
nvmlDeviceValidateInforom	NO	
nvmlDeviceGetDisplayMode	NO	
nvmlDeviceGetDisplayActive	NO	

Function	Migration Support	Diagnostic Message
nvmlDeviceGetPersistenceMode	NO	
nvmlDeviceGetPciInfo	NO	
nvmlDeviceGetMaxPcieLinkGeneration	NO	
nvmlDeviceGetMaxPcieLinkWidth	NO	
nvmlDeviceGetCurrPcieLinkGeneration	NO	
nvmlDeviceGetCurrPcieLinkWidth	NO	
nvmlDeviceGetPcieThroughput	NO	
nvmlDeviceGetPcieReplayCounter	NO	
nvmlDeviceGetClockInfo	NO	
nvmlDeviceGetMaxClockInfo	NO	
nvmlDeviceGetApplicationsClock	NO	
nvmlDeviceGetDefaultApplicationsClock	NO	
nvmlDeviceResetApplicationsClocks	NO	
nvmlDeviceGetClock	NO	
nvmlDeviceGetMaxCustomerBoostClock	NO	
nvmlDeviceGetSupportedMemoryClocks	NO	
nvmlDeviceGetSupportedGraphicsClocks	NO	
nvmlDeviceGetAutoBoostedClocksEnabled	NO	
nvmlDeviceSetAutoBoostedClocksEnabled	NO	
nvmlDeviceSetDefaultAutoBoostedClocksEnabled	NO	
nvmlDeviceGetFanSpeed	NO	
nvmlDeviceGetFanSpeed	NO	
nvmlDeviceGetTemperature	NO	
nvmlDeviceGetTemperatureThreshold	NO	
nvmlDeviceGetPerformanceState	NO	
nvmlDeviceGetCurrentClocksThrottleReasons	NO	
nvmlDeviceGetSupportedClocksThrottleReasons	NO	

Function	Migration Support	Diagnostic Message
nvmlDeviceGetPowerState	NO	
nvmlDeviceGetPowerManagementMode	NO	
nvmlDeviceGetPowerManagementLimit	NO	
nvmlDeviceGetPowerManagementLimitConstraints	NO	
nvmlDeviceGetPowerManagementDefaultLimit	NO	
nvmlDeviceGetPowerUsage	NO	
nvmlDeviceGetTotalEnergyConsumption	NO	
nvmlDeviceGetEnforcedPowerLimit	NO	
nvmlDeviceGetGpuOperationMode	NO	
nvmlDeviceGetMemoryInfo	NO	
nvmlDeviceGetComputeMode	NO	
nvmlDeviceGetCudaComputeCapability	NO	
nvmlDeviceGetEccMode	NO	
nvmlDeviceGetBoardId	NO	
nvmlDeviceGetMultiGpuBoard	NO	
nvmlDeviceGetTotalEccErrors	NO	
nvmlDeviceGetDetailedEccErrors	NO	
nvmlDeviceGetMemoryErrorCounter	NO	
nvmlDeviceGetUtilizationRates	NO	
nvmlDeviceGetEncoderUtilization	NO	
nvmlDeviceGetEncoderCapacity	NO	
nvmlDeviceGetEncoderStats	NO	
nvmlDeviceGetEncoderSessions	NO	
nvmlDeviceGetDecoderUtilization	NO	
nvmlDeviceGetFBCStats	NO	
nvmlDeviceGetFBCSessions	NO	
nvmlDeviceGetDriverModel	NO	
nvmlDeviceGetVbiosVersion	NO	
nvmlDeviceGetBridgeChipInfo	NO	

Function	Migration Support	Diagnostic Message
nvmlDeviceGetComputeRunningProcesses	NO	
nvmlDeviceGetGraphicsRunningProcesses	NO	
nvmlDeviceOnSameBoard	NO	
nvmlDeviceGetAPIRestriction	NO	
nvmlDeviceGetSamples	NO	
nvmlDeviceGetBAR1MemoryInfo	NO	
nvmlDeviceGetViolationStatus	NO	
nvmlDeviceGetRetiredPages	NO	
nvmlDeviceGetRetiredPages	NO	
nvmlDeviceGetRetiredPagesPendingStatus	NO	
nvmlDeviceGetRemappedRows	NO	
nvmlDeviceGetArchitecture	NO	
nvmlDeviceGetMemoryAffinity	NO	
nvmlDeviceGetCpuAffinityWithinScope	NO	
nvmlDeviceGetCpuAffinity	NO	
nvmlDeviceSetCpuAffinity	NO	
nvmlDeviceClearCpuAffinity	NO	
nvmlUnitSetLedState	NO	
nvmlDeviceSetPersistenceMode	NO	
nvmlDeviceSetComputeMode	NO	
nvmlDeviceSetEccMode	NO	
nvmlDeviceClearEccErrorCounts	NO	
nvmlDeviceSetDriverModel	NO	
nvmlDeviceSetGpuLockedClocks	NO	
nvmlDeviceResetGpuLockedClocks	NO	
nvmlDeviceSetApplicationsClocks	NO	
nvmlDeviceSetPowerManagementLimit	NO	
nvmlDeviceSetGpuOperationMode	NO	
nvmlDeviceSetAPIRestriction	NO	

Function	Migration Support	Diagnostic Message
nvmlDeviceGetNvLinkState	NO	
nvmlDeviceGetNvLinkVersion	NO	
nvmlDeviceGetNvLinkCapability	NO	
nvmlDeviceGetNvLinkRemotePciInfo	NO	
nvmlDeviceGetNvLinkErrorCounter	NO	
nvmlDeviceResetNvLinkErrorCounters	NO	
nvmlDeviceSetNvLinkUtilizationControl	NO	
nvmlDeviceGetNvLinkUtilizationControl	NO	
nvmlDeviceGetNvLinkUtilizationCounter	NO	
nvmlDeviceFreezeNvLinkUtilizationCounter	NO	
nvmlDeviceResetNvLinkUtilizationCounter	NO	
nvmlEventSetCreate	NO	
nvmlDeviceRegisterEvents	NO	
nvmlDeviceGetSupportedEventTypes	NO	
nvmlEventSetWait	NO	
nvmlEventSetFree	NO	
nvmlDeviceModifyDrainState	NO	
nvmlDeviceQueryDrainState	NO	
nvmlDeviceRemoveGpu	NO	
nvmlDeviceDiscoverGpus	NO	
nvmlDeviceGetFieldValues	NO	
nvmlDeviceGetVirtualizationMode	NO	
nvmlDeviceGetHostVgpuMode	NO	
nvmlDeviceSetVirtualizationMode	NO	
nvmlDeviceGetGridLicensableFeatures	NO	
nvmlDeviceGetProcessUtilization	NO	
nvmlDeviceGetSupportedVgpus	NO	
nvmlDeviceGetCreatableVgpus	NO	
nvmlVgpuTypeGetClass	NO	

Function	Migration Support	Diagnostic Message
nvmlVgpuTypeGetName	NO	
nvmlVgpuTypeGetDeviceID	NO	
nvmlVgpuTypeGetFramebufferSize	NO	
nvmlVgpuTypeGetNumDisplayHeads	NO	
nvmlVgpuTypeGetResolution	NO	
nvmlVgpuTypeGetLicense	NO	
nvmlVgpuTypeGetFrameRateLimit	NO	
nvmlVgpuTypeGetMaxInstances	NO	
nvmlVgpuTypeGetMaxInstancesPerVm	NO	
nvmlDeviceGetActiveVgpus	NO	
nvmlVgpuInstanceGetVmID	NO	
nvmlVgpuInstanceGetUUID	NO	
nvmlVgpuInstanceGetVmDriverVersion	NO	
nvmlVgpuInstanceGetFbUsage	NO	
nvmlVgpuInstanceGetLicenseStatus	NO	
nvmlVgpuInstanceGetType	NO	
nvmlVgpuInstanceGetFrameRateLimit	NO	
nvmlVgpuInstanceGetEccMode	NO	
nvmlVgpuInstanceGetEncoderCapacity	NO	
nvmlVgpuInstanceSetEncoderCapacity	NO	
nvmlVgpuInstanceGetEncoderStats	NO	
nvmlVgpuInstanceGetEncoderSessions	NO	
nvmlVgpuInstanceGetFBCStats	NO	
nvmlVgpuInstanceGetFBCSessions	NO	
nvmlVgpuInstanceGetMetadata	NO	
nvmlDeviceGetVgpuMetadata	NO	
nvmlGetVgpuCompatibility	NO	
nvmlDeviceGetPgpuMetadataString	NO	
nvmlGetVgpuVersion	NO	
nvmlSetVgpuVersion	NO	

Function	Migration Support	Diagnostic Message
nvmlDeviceGetVgpuUtilization	NO	
nvmlDeviceGetVgpuProcessUtilization	NO	
nvmlVgpuInstanceGetAccountingMode	NO	
nvmlVgpuInstanceGetAccountingPids	NO	
nvmlVgpuInstanceGetAccountingStats	NO	
nvmlVgpuInstanceClearAccountingPids	NO	
nvmlGetBlacklistDeviceCount	NO	
nvmlGetBlacklistDeviceInfoByIndex	NO	
nvmlDeviceSetMigMode	NO	
nvmlDeviceGetMigMode	NO	
nvmlDeviceGetGpuInstanceProfileInfo	NO	
nvmlDeviceGetGpuInstancePossiblePlacements	NO	
nvmlDeviceGetGpuInstanceRemainingCapacity	NO	
nvmlDeviceCreateGpuInstance	NO	
nvmlGpuInstanceDestroy	NO	
nvmlDeviceGetGpuInstances	NO	
nvmlDeviceGetGpuInstanceById	NO	
nvmlGpuInstanceGetInfo	NO	
nvmlGpuInstanceGetComputeInstanceProfileInfo	NO	
nvmlGpuInstanceGetComputeInstanceRemainingCapacity	NO	
nvmlGpuInstanceCreateComputeInstance	NO	
nvmlComputeInstanceDestroy	NO	
nvmlGpuInstanceGetComputeInstances	NO	
nvmlGpuInstanceGetComputeInstanceById	NO	
nvmlComputeInstanceGetInfo	NO	
nvmlDeviceIsMigDeviceHandle	NO	
nvmlDeviceGetGpuInstanceId	NO	

Function	Migration Support	Diagnostic Message
nvmlDeviceGetComputeInstanceId	NO	
nvmlDeviceGetMaxMigDeviceCount	NO	
nvmlDeviceGetMigDeviceHandleByIndex	NO	
nvmlDeviceGetDeviceHandleFromMigDeviceHandle	NO	

Runtime and Driver API

Function	Migration Support	Diagnostic Message
cudaChooseDevice	NO	
cudaDeviceFlushGPUDirectRDMAWrites	NO	
cudaDeviceGetAttribute	YES	DPCT1076
cudaDeviceGetByPCIBusId	NO	
cudaDeviceGetCacheConfig	YES	DPCT1026 / DPCT1027
cudaDeviceGetDefaultMemPool	NO	
cudaDeviceGetLimit	YES	DPCT1029
cudaDeviceGetMemPool	NO	
cudaDeviceGetNvSciSyncAttributes	NO	
cudaDeviceGetP2PAttribute	NO	
cudaDeviceGetPCIBusId	NO	
cudaDeviceGetSharedMemConfig	NO	
cudaDeviceGetStreamPriorityRange	YES	DPCT1026 / DPCT1027
cudaDeviceGetTexture1DLinearMaxWidth	NO	
cudaDeviceReset	YES	
cudaDeviceSetCacheConfig	YES	DPCT1026 / DPCT1027
cudaDeviceSetLimit	YES	DPCT1026 / DPCT1027
cudaDeviceSetMemPool	NO	
cudaDeviceSetSharedMemConfig	YES	DPCT1026 / DPCT1027
cudaDeviceSynchronize	YES	
cudaDeviceUnregisterAsyncNotification	NO	
cudaDeviceRegisterAsyncNotification	NO	

Function	Migration Support	Diagnostic Message
cudaGetDevice	YES	
cudaGetDeviceCount	YES	
cudaGetDeviceFlags	NO	
cudaGetDeviceProperties	YES	
cudaGetDeviceProperties_v2	YES	
cudaInitDevice	NO	
cudaIpcCloseMemHandle	NO	
cudaIpcGetEventHandle	NO	
cudaIpcGetMemHandle	NO	
cudaIpcOpenEventHandle	NO	
cudaIpcOpenMemHandle	NO	
cudaSetDevice	YES	DPCT1093
cudaSetDeviceFlags	YES	DPCT1026 / DPCT1027
cudaSetValidDevices	NO	
cudaThreadExit	YES	
cudaThreadGetCacheConfig	NO	
cudaThreadGetLimit	NO	
cudaThreadSetCacheConfig	NO	
cudaThreadSetLimit	YES	DPCT1026 / DPCT1027
cudaThreadSynchronize	YES	
cudaGetErrorName	YES	
cudaGetErrorString	YES	
cudaGetLastError	YES	DPCT1010
cudaPeekAtLastError	YES	DPCT1026 / DPCT1027
cudaCtxResetPersistingL2Cache	YES	DPCT1026 / DPCT1027
cudaStreamAddCallback	YES	
cudaStreamAttachMemAsync	YES	DPCT1026 / DPCT1027
cudaStreamBeginCapture	YES	DPCT1119
cudaStreamBeginCaptureToGraph	NO	
cudaStreamCopyAttributes	NO	

Function	Migration Support	Diagnostic Message
cudaStreamCreate	YES	
cudaStreamCreateWithFlags	YES	DPCT1025
cudaStreamCreateWithPriority	YES	DPCT1025
cudaStreamDestroy	YES	
cudaStreamEndCapture	YES	DPCT1119
cudaStreamGetAttribute	YES	DPCT1026 / DPCT1027
cudaStreamGetCaptureInfo	NO	
cudaStreamGetCaptureInfo_v3	NO	
cudaStreamGetFlags	YES	DPCT1014
cudaStreamGetId	NO	
cudaStreamGetPriority	YES	DPCT1014
cudaStreamIsCapturing	YES	DPCT1119
cudaStreamQuery	YES	DPCT1026 / DPCT1027
cudaStreamSetAttribute	YES	DPCT1026 / DPCT1027
cudaStreamSynchronize	YES	
cudaStreamUpdateCaptureDependencies	NO	
cudaStreamUpdateCaptureDependencies_v2	NO	
cudaStreamWaitEvent	YES	
cudaThreadExchangeStreamCaptureMode	NO	
cudaEventCreate	YES	DPCT1026 / DPCT1027
cudaEventCreateWithFlags	YES	DPCT1026 / DPCT1027
cudaEventDestroy	YES	DPCT1026 / DPCT1027
cudaEventElapsedTime	YES	DPCT1012
cudaEventQuery	YES	
cudaEventRecord	YES	DPCT1012 / DPCT1024
cudaEventRecordWithFlags	NO	
cudaEventSynchronize	YES	DPCT1012
cudaDestroyExternalMemory	NO	
cudaDestroyExternalSemaphore	NO	

Function	Migration Support	Diagnostic Message
cudaExternalMemoryGetMappedBuffer	NO	
cudaExternalMemoryGetMappedMipmappedArray	NO	
cudaImportExternalMemory	NO	
cudaImportExternalSemaphore	NO	
cudaSignalExternalSemaphoresAsync	NO	
cudaWaitExternalSemaphoresAsync	NO	
cudaFuncGetAttributes	YES	
cudaFuncGetName	NO	
cudaFuncSetAttribute	YES	DPCT1026 / DPCT1027
cudaFuncSetCacheConfig	YES	DPCT1026
cudaFuncSetSharedMemConfig	YES	DPCT1026
cudaGetParameterBuffer	NO	
cudaGetParameterBufferV2	NO	
cudaFuncGetParamInfo	NO	
cudaGridDependencySynchronize	NO	
cudaLaunchCooperativeKernel	YES	Partial
cudaLaunchCooperativeKernelMultiDevice	NO	
cudaLaunchDevice	NO	
cudaLaunchHostFunc	NO	
cudaLaunchKernel	YES	Partial
cudaLaunchKernelExC	NO	
cudaSetDoubleForDevice	NO	
cudaSetDoubleForHost	NO	
cudaTriggerProgrammaticLaunchCompletion	NO	
cudaOccupancyAvailableDynamicSMemPerBlock	NO	
cudaOccupancyMaxActiveBlocksPerMultiprocessor	YES	DPCT1111
cudaOccupancyMaxActiveBlocksPerMultiprocessorWithFlags	NO	

Function	Migration Support	Diagnostic Message
cudaOccupancyMaxActiveClusters	NO	
cudaOccupancyMaxPotentialClusterSize	NO	
cudaConfigureCall	NO	
cudaLaunch	NO	
cudaSetupArgument	NO	
cudaArrayGetInfo	YES	
cudaArrayGetMemoryRequirements	NO	
cudaArrayGetPlane	NO	
cudaArrayGetSparseProperties	NO	
cudaFree	YES	
cudaFreeArray	YES	
cudaFreeHost	YES	
cudaFreeMipmappedArray	YES	DPCT1026 / DPCT1027
cudaGetMipmappedArrayLevel	YES	DPCT1026 / DPCT1027
cudaGetSymbolAddress	YES	
cudaGetSymbolSize	YES	
cudaHostAlloc	YES	
cudaHostGetDevicePointer	YES	Partial
cudaHostGetFlags	YES	
cudaHostRegister	YES	DPCT1026 / DPCT1027
cudaHostUnregister	YES	DPCT1026 / DPCT1027
cudaMalloc	YES	
cudaMalloc3D	YES	
cudaMalloc3DArray	YES	
cudaMallocArray	YES	
cudaMallocHost	YES	
cudaMallocManaged	YES	
cudaMallocMipmappedArray	YES	DPCT1026 / DPCT1027
cudaMallocPitch	YES	
cudaMemAdvise	YES	DPCT1063 Partial

Function	Migration Support	Diagnostic Message
cudaMemAdvise_v2	NO	
cudaMemGetInfo	YES	DPCT1072 / DPCT1106
cudaMemPrefetchAsync	YES	Partial
cudaMemRangeGetAttribute	NO	
cudaMemRangeGetAttributes	NO	
cudaMemcpy	YES	
cudaMemcpy2D	YES	
cudaMemcpy2DArrayToArray	YES	
cudaMemcpy2DAsync	YES	
cudaMemcpy2DFromArray	YES	
cudaMemcpy2DFromArrayAsync	YES	
cudaMemcpy2DToArray	YES	
cudaMemcpy2DToArrayAsync	YES	
cudaMemcpy3D	YES	
cudaMemcpy3DAsync	YES	
cudaMemcpy3DPeer	YES	
cudaMemcpy3DPeerAsync	YES	
cudaMemcpyAsync	YES	
cudaMemcpyFromSymbol	YES	
cudaMemcpyFromSymbolAsync	YES	
cudaMemcpyPeer	YES	
cudaMemcpyPeerAsync	YES	
cudaMemcpyToSymbol	YES	
cudaMemcpyToSymbolAsync	YES	
cudaMemset	YES	
cudaMemset2D	YES	
cudaMemset2DAsync	YES	
cudaMemset3D	YES	
cudaMemset3DAsync	YES	
cudaMemsetAsync	YES	

Function	Migration Support	Diagnostic Message
cudaMipmappedArrayGetMemoryRequirements	NO	
cudaMipmappedArrayGetSparseProperties	NO	
make_cudaExtent	YES	
make_cudaPitchedPtr	YES	
make_cudaPos	YES	
cudaMemcpyArrayToArray	YES	
cudaMemcpyFromArray	YES	
cudaMemcpyFromArrayAsync	YES	
cudaMemcpyToArray	YES	
cudaMemcpyToArrayAsync	YES	
cudaFreeAsync	NO	
cudaMallocAsync	NO	
cudaMallocFromPoolAsync	NO	
cudaMemPoolCreate	NO	
cudaMemPoolDestroy	NO	
cudaMemPoolExportPointer	NO	
cudaMemPoolExportToShareableHandle	NO	
cudaMemPoolGetAccess	NO	
cudaMemPoolGetAttribute	NO	
cudaMemPoolImportFromShareableHandle	NO	
cudaMemPoolImportPointer	NO	
cudaMemPoolSetAccess	NO	
cudaMemPoolSetAttribute	NO	
cudaMemPoolTrimTo	NO	
cudaPointerGetAttributes	YES	
cudaDeviceCanAccessPeer	YES	DPCT1031
cudaDeviceDisablePeerAccess	YES	DPCT1026 / DPCT1027
cudaDeviceEnablePeerAccess	YES	DPCT1026 / DPCT1027
cudaGLGetDevices	NO	

Function	Migration Support	Diagnostic Message
cudaGraphicsGLRegisterBuffer	NO	
cudaGraphicsGLRegisterImage	NO	
cudaWGLGetDevice	NO	
cudaGLMapBufferObject	NO	
cudaGLMapBufferObjectAsync	NO	
cudaGLRegisterBufferObject	NO	
cudaGLSetBufferObjectMapFlags	NO	
cudaGLSetGLDevice	NO	
cudaGLUnmapBufferObject	NO	
cudaGLUnmapBufferObjectAsync	NO	
cudaGLUnregisterBufferObject	NO	
cudaD3D9GetDevice	NO	
cudaD3D9GetDevices	NO	
cudaD3D9GetDirect3DDevice	NO	
cudaD3D9SetDirect3DDevice	NO	
cudaGraphicsD3D9RegisterResource	NO	
cudaD3D9MapResources	NO	
cudaD3D9RegisterResource	NO	
cudaD3D9ResourceGetMappedArray	NO	
cudaD3D9ResourceGetMappedPitch	NO	
cudaD3D9ResourceGetMappedPointer	NO	
cudaD3D9ResourceGetMappedSize	NO	
cudaD3D9ResourceGetSurfaceDimensions	NO	
cudaD3D9ResourceSetMapFlags	NO	
cudaD3D9UnmapResources	NO	
cudaD3D9UnregisterResource	NO	
cudaD3D10GetDevice	NO	
cudaD3D10GetDevices	NO	
cudaGraphicsD3D10RegisterResource	NO	
cudaD3D10GetDirect3DDevice	NO	

Function	Migration Support	Diagnostic Message
cudaD3D10MapResources	NO	
cudaD3D10RegisterResource	NO	
cudaD3D10ResourceGetMappedArray	NO	
cudaD3D10ResourceGetMappedPitch	NO	
cudaD3D10ResourceGetMappedPointer	NO	
cudaD3D10ResourceGetMappedSize	NO	
cudaD3D10ResourceGetSurfaceDimensions	NO	
cudaD3D10ResourceSetMapFlags	NO	
cudaD3D10SetDirect3DDevice	NO	
cudaD3D10UnmapResources	NO	
cudaD3D10UnregisterResource	NO	
cudaD3D11GetDevice	NO	
cudaD3D11GetDevices	NO	
cudaGraphicsD3D11RegisterResource	YES	DPCT1119
cudaD3D11GetDirect3DDevice	NO	
cudaD3D11SetDirect3DDevice	NO	
cudaGraphicsVDPAURegisterOutputSurface	NO	
cudaGraphicsVDPAURegisterVideoSurface	NO	
cudaVDPAUGetDevice	NO	
cudaVDPAUSetVDPAUDevice	NO	
cudaEGLStreamConsumerAcquireFrame	NO	
cudaEGLStreamConsumerConnect	NO	
cudaEGLStreamConsumerConnectWithFlags	NO	
cudaEGLStreamConsumerDisconnect	NO	
cudaEGLStreamConsumerReleaseFrame	NO	
cudaEGLStreamProducerConnect	NO	
cudaEGLStreamProducerDisconnect	NO	
cudaEGLStreamProducerPresentFrame	NO	

Function	Migration Support	Diagnostic Message
cudaEGLStreamProducerReturnFrame	NO	
cudaEventCreateFromEGLSync	NO	
cudaGraphicsEGLRegisterImage	NO	
cudaGraphicsResourceGetMappedEglFrame	NO	
cudaGraphicsMapResources	YES	DPCT1119
cudaGraphicsResourceGetMappedMipmappedArray	YES	DPCT1119
cudaGraphicsResourceGetMappedPointer	YES	DPCT1119
cudaGraphicsResourceSetMapFlags	YES	DPCT1026 / DPCT1027
cudaGraphicsSubResourceGetMappedArray	YES	DPCT1119
cudaGraphicsUnmapResources	YES	DPCT1119
cudaGraphicsUnregisterResource	YES	DPCT1119
cudaBindTexture	YES	
cudaBindTexture2D	YES	
cudaBindTextureToArray	YES	
cudaBindTextureToMipmappedArray	YES	
cudaGetTextureAlignmentOffset	NO	
cudaGetTextureReference	NO	
cudaUnbindTexture	YES	
cudaBindSurfaceToArray	NO	
cudaGetSurfaceReference	NO	
cudaCreateChannelDesc	YES	
cudaCreateChannelDescHalf	YES	
cudaCreateChannelDescHalf1	YES	
cudaCreateChannelDescHalf2	YES	
cudaCreateChannelDescHalf4	YES	
cudaCreateTextureObject	YES	
cudaDestroyTextureObject	YES	
cudaGetChannelDesc	YES	
cudaGetTextureObjectResourceDesc	YES	

Function	Migration Support	Diagnostic Message
cudaGetTextureObjectResourceViewDesc	NO	
cudaGetTextureObjectTextureDesc	YES	
cudaCreateSurfaceObject	NO	
cudaDestroySurfaceObject	NO	
cudaGetSurfaceObjectResourceDesc	NO	
cudaDriverGetVersion	YES	DPCT1043
cudaRuntimeGetVersion	YES	DPCT1043
cudaGraph_t	YES	DPCT1119
cudaGraphExec_t	YES	DPCT1119
cudaGraphNode_t	YES	DPCT1119
cudaDeviceGetGraphMemAttribute	NO	
cudaDeviceGraphMemTrim	NO	
cudaDeviceSetGraphMemAttribute	NO	
cudaGetCurrentGraphExec	NO	
cudaGraphAddChildGraphNode	NO	
cudaGraphAddDependencies	YES	DPCT1119
cudaGraphAddDependencies_v2	NO	
cudaGraphAddEmptyNode	YES	DPCT1119
cudaGraphAddEventRecordNode	NO	
cudaGraphAddEventWaitNode	NO	
cudaGraphAddExternalSemaphoresSignalNode	NO	
cudaGraphAddExternalSemaphoresWaitNode	NO	
cudaGraphAddHostNode	NO	
cudaGraphAddKernelNode	NO	
cudaGraphAddMemAllocNode	NO	
cudaGraphAddMemFreeNode	NO	
cudaGraphAddMemcpyNode	NO	
cudaGraphAddMemcpyNode1D	NO	
cudaGraphAddMemcpyNodeFromSymbol	NO	

Function	Migration Support	Diagnostic Message
cudaGraphAddMemcpyNodeToSymbol	NO	
cudaGraphAddMemsetNode	NO	
cudaGraphAddNode	NO	
cudaGraphAddNode_v2	NO	
cudaGraphChildGraphNodeGetGraph	NO	
cudaGraphClone	NO	
cudaGraphConditionalHandleCreate	NO	
cudaGraphCreate	NO	
cudaGraphDebugDotPrint	NO	
cudaGraphDestroy	NO	
cudaGraphDestroyNode	NO	
cudaGraphEventRecordNodeGetEvent	NO	
cudaGraphEventRecordNodeSetEvent	NO	
cudaGraphEventWaitNodeGetEvent	NO	
cudaGraphEventWaitNodeSetEvent	NO	
cudaGraphExecChildGraphNodeSetParams	NO	
cudaGraphExecDestroy	YES	DPCT1119
cudaGraphExecEventRecordNodeSetEvent	NO	
cudaGraphExecEventWaitNodeSetEvent	NO	
cudaGraphExecExternalSemaphoresSignalNodeSetParams	NO	
cudaGraphExecExternalSemaphoresWaitNodeSetParams	NO	
cudaGraphExecGetFlags	NO	
cudaGraphExecHostNodeSetParams	NO	
cudaGraphExecKernelNodeSetParams	NO	
cudaGraphExecMemcpyNodeSetParams	NO	
cudaGraphExecMemcpyNodeSetParams1D	NO	
cudaGraphExecMemcpyNodeSetParamsFromSymbol	NO	

Function	Migration Support	Diagnostic Message
cudaGraphExecMemcpyNodeSetParamsToSymbol	NO	
cudaGraphExecMemsetNodeSetParams	NO	
cudaGraphExecNodeSetParams	NO	
cudaGraphExecUpdate	YES	DPCT1119
cudaGraphExternalSemaphoresSignalNodeGetParams	NO	
cudaGraphExternalSemaphoresSignalNodeSetParams	NO	
cudaGraphExternalSemaphoresWaitNodeGetParams	NO	
cudaGraphExternalSemaphoresWaitNodeSetParams	NO	
cudaGraphGetEdges	NO	
cudaGraphGetEdges_v2	NO	
cudaGraphGetNodes	NO	
cudaGraphGetRootNodes	NO	
cudaGraphHostNodeGetParams	NO	
cudaGraphHostNodeSetParams	NO	
cudaGraphInstantiate	YES	DPCT1119
cudaGraphInstantiateWithFlags	NO	
cudaGraphInstantiateWithParams	NO	
cudaGraphKernelNodeCopyAttributes	NO	
cudaGraphKernelNodeGetAttribute	NO	
cudaGraphKernelNodeGetParams	NO	
cudaGraphKernelNodeSetAttribute	NO	
cudaGraphKernelNodeSetEnabled	NO	
cudaGraphKernelNodeSetGridDim	NO	
cudaGraphKernelNodeSetParam	NO	
cudaGraphKernelNodeSetParams	NO	
cudaGraphKernelNodeUpdatesApply	NO	
cudaGraphLaunch	YES	DPCT1119

Function	Migration Support	Diagnostic Message
cudaGraphMemAllocNodeGetParams	NO	
cudaGraphMemFreeNodeGetParams	NO	
cudaGraphMemcpyNodeGetParams	NO	
cudaGraphMemcpyNodeSetParams	NO	
cudaGraphMemcpyNodeSetParams1D	NO	
cudaGraphMemcpyNodeSetParamsFromSymbol	NO	
cudaGraphMemcpyNodeSetParamsToSymbol	NO	
cudaGraphMemsetNodeGetParams	NO	
cudaGraphMemsetNodeSetParams	NO	
cudaGraphNodeFindInClone	NO	
cudaGraphNodeGetDependencies	NO	
cudaGraphNodeGetDependencies_v2	NO	
cudaGraphNodeGetDependentNodes	NO	
cudaGraphNodeGetDependentNodes_v2	NO	
cudaGraphNodeGetEnabled	NO	
cudaGraphNodeGetType	NO	
cudaGraphNodeSetEnabled	NO	
cudaGraphNodeSetParams	NO	
cudaGraphReleaseUserObject	NO	
cudaGraphRemoveDependencies	NO	
cudaGraphRemoveDependencies_v2	NO	
cudaGraphRetainUserObject	NO	
cudaGraphSetConditional	NO	
cudaGraphKernelNodeSetGridDim	NO	
cudaGraphKernelNodeUpdatesApply	NO	
cudaGraphKernelNodeSetEnabled	NO	
cudaGraphConditionalHandleCreate	NO	
cudaGraphKernelNodeSetParam	NO	
cudaGraphUpload	NO	

Function	Migration Support	Diagnostic Message
cudaUserObjectCreate	NO	
cudaUserObjectRelease	NO	
cudaUserObjectRetain	NO	
cudaGetDriverEntryPoint	NO	
cudaGetDriverEntryPointByVersion	NO	
cudaOccupancyMaxPotentialBlockSize	YES	DPCT1111
cudaOccupancyMaxPotentialBlockSizeVariableSMem	NO	
cudaOccupancyMaxPotentialBlockSizeVariableSMemWithFlags	NO	
cudaOccupancyMaxPotentialBlockSizeWithFlags	NO	
cudaGetFuncBySymbol	NO	
cudaGetKernel	NO	
cudaProfilerInitialize	NO	
cudaProfilerStart	YES	DPCT1026 / DPCT1027
cudaProfilerStop	YES	DPCT1026 / DPCT1027
atomicAdd	YES	DPCT1039
atomicSub	YES	DPCT1039
atomicExch	YES	DPCT1039
atomicMin	YES	DPCT1039
atomicMax	YES	DPCT1039
atomicInc	YES	DPCT1039
atomicDec	YES	DPCT1039
atomicCAS	YES	DPCT1039
atomicAnd	YES	DPCT1039
atomicOr	YES	DPCT1039
atomicXor	YES	DPCT1039
__nv_cvt_bfloat16raw2_to_fp8x2	NO	
__nv_cvt_bfloat16raw_to_fp8	NO	
__nv_cvt_double2_to_fp8x2	NO	
__nv_cvt_double_to_fp8	NO	

Function	Migration Support	Diagnostic Message
__nv_cvt_float2_to_fp8x2	NO	
__nv_cvt_float_to_fp8	NO	
__nv_cvt_fp8_to_halfraw	NO	
__nv_cvt_fp8x2_to_halfraw2	NO	
__nv_cvt_halfraw2_to_fp8x2	NO	
__nv_cvt_halfraw_to_fp8	NO	
__habs	YES	
__hadd	YES	
__hadd_rn	YES	
__hadd_sat	YES	
__hdiv	YES	
__hfma	YES	
__hfma_relu	YES	
__hfma_sat	YES	
__hmul	YES	
__hmul_rn	YES	
__hmul_sat	YES	
__hneg	YES	
__hsub	YES	
__hsub_rn	YES	
__hsub_sat	YES	
__h2div	YES	
__habs2	YES	
__hadd2	YES	
__hadd2_rn	YES	
__hadd2_sat	YES	
__hcmadd	YES	
__hfma2	YES	
__hfma2_relu	YES	
__hfma2_sat	YES	

Function	Migration Support	Diagnostic Message
__hmul2	YES	
__hmul2_rn	YES	
__hmul2_sat	YES	
__hneg2	YES	
__hsub2	YES	
__hsub2_rn	YES	
__hsub2_sat	YES	
__heq	YES	
__hequ	YES	
__hge	YES	
__hgeu	YES	
__hgt	YES	
__hgtu	YES	
__hisinf	YES	
__hisnan	YES	
__hle	YES	
__hleu	YES	
__hlt	YES	
__hltu	YES	
__hmax	YES	
__hmax_nan	YES	
__hmin	YES	
__hmin_nan	YES	
__hne	YES	
__hneu	YES	
__hbeq2	YES	
__hbequ2	YES	
__hbge2	YES	
__hbgeu2	YES	
__hbgt2	YES	

Function	Migration Support	Diagnostic Message
__hbgtu2	YES	
__hble2	YES	
__hbleu2	YES	
__hblt2	YES	
__hbltu2	YES	
__hbne2	YES	
__hbneu2	YES	
__heq2	YES	
__heq2_mask	YES	
__hequ2	YES	
__hequ2_mask	YES	
__hge2	YES	
__hge2_mask	YES	
__hgeu2	YES	
__hgeu2_mask	YES	
__hgt2	YES	
__hgt2_mask	YES	
__hgtu2	YES	
__hgtu2_mask	YES	
__hisnan2	YES	
__hle2	YES	
__hle2_mask	YES	
__hleu2	YES	
__hleu2_mask	YES	
__hlt2	YES	
__hlt2_mask	YES	
__hltu2	YES	
__hltu2_mask	YES	
__hmax2	YES	
__hmax2_nan	YES	

Function	Migration Support	Diagnostic Message
__hmin2	YES	
__hmin2_nan	YES	
__hne2	YES	
__hne2_mask	YES	
__hneu2	YES	
__hneu2_mask	YES	
__double2half	YES	
__float22half2_rn	YES	
__float2half	YES	
__float2half2_rn	YES	
__float2half_rd	YES	
__float2half_rn	YES	
__float2half_ru	YES	
__float2half_rz	YES	
__floats2half2_rn	YES	
__half22float2	YES	
__half2char_rz	NO	
__half2float	YES	
__half2half2	YES	
__half2int_rd	YES	
__half2int_rn	YES	
__half2int_ru	YES	
__half2int_rz	YES	
__half2ll_rd	YES	
__half2ll_rn	YES	
__half2ll_ru	YES	
__half2ll_rz	YES	
__half2short_rd	YES	
__half2short_rn	YES	
__half2short_ru	YES	

Function	Migration Support	Diagnostic Message
__half2short_rz	YES	
__half2uchar_rz	NO	
__half2uint_rd	YES	
__half2uint_rn	YES	
__half2uint_ru	YES	
__half2uint_rz	YES	
__half2ull_rd	YES	
__half2ull_rn	YES	
__half2ull_ru	YES	
__half2ull_rz	YES	
__half2ushort_rd	YES	
__half2ushort_rn	YES	
__half2ushort_ru	YES	
__half2ushort_rz	YES	
__half_as_short	YES	
__half_as_ushort	YES	
__halves2half2	YES	
__high2float	YES	
__high2half	YES	
__high2half2	YES	
__highs2half2	YES	
__int2half_rd	YES	
__int2half_rn	YES	
__int2half_ru	YES	
__int2half_rz	YES	
__ldca	YES	DPCT1098
__ldcg	YES	DPCT1098
__ldcs	YES	DPCT1098
__ldcv	YES	DPCT1098
__ldg	YES	DPCT1098

Function	Migration Support	Diagnostic Message
__ldlu	YES	DPCT1098
__ll2half_rd	YES	
__ll2half_rn	YES	
__ll2half_ru	YES	
__ll2half_rz	YES	
__low2float	YES	
__low2half	YES	
__low2half2	YES	
__lowhigh2highlow	YES	
__lows2half2	YES	
__shfl_down_sync	YES	
__shfl_sync	YES	
__shfl_up_sync	YES	
__shfl_xor_sync	YES	
__short2half_rd	YES	
__short2half_rn	YES	
__short2half_ru	YES	
__short2half_rz	YES	
__short_as_half	YES	
__stcg	YES	DPCT1098
__stcs	YES	DPCT1098
__stwb	YES	DPCT1098
__stwt	YES	DPCT1098
__uint2half_rd	YES	
__uint2half_rn	YES	
__uint2half_ru	YES	
__uint2half_rz	YES	
__ull2half_rd	YES	
__ull2half_rn	YES	
__ull2half_ru	YES	

Function	Migration Support	Diagnostic Message
__ull2half_rz	YES	
__ushort2half_rd	YES	
__ushort2half_rn	YES	
__ushort2half_ru	YES	
__ushort2half_rz	YES	
__ushort_as_half	YES	
make_half2	YES	
hceil	YES	
hcos	YES	
hexp	YES	
hexp10	YES	
hexp2	YES	
hfloor	YES	
hlog	YES	
hlog10	YES	
hlog2	YES	
hrcp	YES	
hrint	YES	
hrsqrt	YES	
hsin	YES	
hsqrt	YES	
htrunc	YES	
h2ceil	YES	
h2cos	YES	
h2exp	YES	
h2exp10	YES	
h2exp2	YES	
h2floor	YES	
h2log	YES	
h2log10	YES	

Function	Migration Support	Diagnostic Message
h2log2	YES	
h2rcp	YES	
h2rint	YES	
h2rsqrt	YES	
h2sin	YES	
h2sqrt	YES	
h2trunc	YES	
__bfloat1622float2	YES	
__bfloat162bfloat162	YES	
__bfloat162char_rz	NO	
__bfloat162float	YES	
__bfloat162int_rd	YES	
__bfloat162int_rn	YES	
__bfloat162int_ru	YES	
__bfloat162int_rz	YES	
__bfloat162ll_rd	YES	
__bfloat162ll_rn	YES	
__bfloat162ll_ru	YES	
__bfloat162ll_rz	YES	
__bfloat162short_rd	YES	
__bfloat162short_rn	YES	
__bfloat162short_ru	YES	
__bfloat162short_rz	YES	
__bfloat162uchar_rz	NO	
__bfloat162uint_rd	YES	
__bfloat162uint_rn	YES	
__bfloat162uint_ru	YES	
__bfloat162uint_rz	YES	
__bfloat162ull_rd	YES	
__bfloat162ull_rn	YES	

Function	Migration Support	Diagnostic Message
__bfloat162ull_ru	YES	
__bfloat162ull_rz	YES	
__bfloat162ushort_rd	YES	
__bfloat162ushort_rn	YES	
__bfloat162ushort_ru	YES	
__bfloat162ushort_rz	YES	
__bfloat16_as_short	YES	
__bfloat16_as_ushort	YES	
__double2bfloat16	YES	
__float22bfloat162_rn	YES	
__float2bfloat16	YES	
__float2bfloat162_rn	YES	
__float2bfloat16_rd	YES	
__float2bfloat16_rn	YES	
__float2bfloat16_ru	YES	
__float2bfloat16_rz	YES	
__floats2bfloat162_rn	YES	
__halves2bfloat162	YES	
__high2bfloat16	YES	
__high2bfloat162	YES	
__highs2bfloat162	YES	
__int2bfloat16_rd	YES	
__int2bfloat16_rn	YES	
__int2bfloat16_ru	YES	
__int2bfloat16_rz	YES	
__ll2bfloat16_rd	YES	
__ll2bfloat16_rn	YES	
__ll2bfloat16_ru	YES	
__ll2bfloat16_rz	YES	
__low2bfloat16	YES	

Function	Migration Support	Diagnostic Message
__low2bfloat162	YES	
__lows2bfloat162	YES	
__short2bfloat16_rd	YES	
__short2bfloat16_rn	YES	
__short2bfloat16_ru	YES	
__short2bfloat16_rz	YES	
__short_as_bfloat16	YES	
__uint2bfloat16_rd	YES	
__uint2bfloat16_rn	YES	
__uint2bfloat16_ru	YES	
__uint2bfloat16_rz	YES	
__ull2bfloat16_rd	YES	
__ull2bfloat16_rn	YES	
__ull2bfloat16_ru	YES	
__ull2bfloat16_rz	YES	
__ushort2bfloat16_rd	YES	
__ushort2bfloat16_rn	YES	
__ushort2bfloat16_ru	YES	
__ushort2bfloat16_rz	YES	
__ushort_as_bfloat16	YES	
make_bfloat162	NO	
acosf	YES	
acoshf	YES	
asinf	YES	
asinhf	YES	
atan2f	YES	
atanf	YES	
atanhf	YES	
cbrtf	YES	
ceilf	YES	

Function	Migration Support	Diagnostic Message
copysignf	YES	
cosf	YES	
coshf	YES	
cospif	YES	
cyl_bessel_i0f	YES	
cyl_bessel_i1f	YES	
erfcf	YES	
erfcinvf	YES	
erfcxf	YES	DPCT1017
erff	YES	
erfinvf	YES	
exp10f	YES	
exp2f	YES	
expf	YES	
expm1f	YES	
fabsf	YES	
fdimf	YES	
fdividef	YES	
floorf	YES	
fmaf	YES	
fmaxf	YES	
fminf	YES	
fmodf	YES	
frexpf	YES	DPCT1017
hypotf	YES	
ilogbf	YES	
isfinite	YES	
isinf	YES	
isnan	YES	
j0f	YES	

Function	Migration Support	Diagnostic Message
j1f	YES	
jnf	YES	
ldexpf	YES	
lgammaf	YES	
llrintf	YES	
llroundf	YES	
log10f	YES	
log1pf	YES	
log2f	YES	
logbf	YES	
logf	YES	
lrintf	YES	
lroundf	YES	
max	YES	
min	YES	
modff	YES	DPCT1017
nanf	YES	DPCT1017
nearbyintf	YES	DPCT1017
nextafterf	YES	
norm3df	YES	
norm4df	YES	
normcdff	YES	
normcdfinvf	YES	
normf	YES	DPCT1017
powf	YES	
rcbrtf	YES	DPCT1017
remainderf	YES	
remquof	YES	DPCT1017
rhypotf	YES	DPCT1017
rintf	YES	

Function	Migration Support	Diagnostic Message
rnorm3df	YES	
rnorm4df	YES	
rnormf	YES	DPCT1017
roundf	YES	
rsqrtf	YES	
scalblnf	YES	DPCT1017
scalbnf	YES	DPCT1017
signbit	YES	
sincosf	YES	DPCT1017
sincospif	YES	DPCT1017
sinf	YES	
sinhf	YES	
sinpif	YES	
sqrtf	YES	
tanf	YES	
tanhf	YES	
tgammaf	YES	
truncf	YES	
y0f	YES	
y1f	YES	
ynf	YES	
acos	YES	
acosh	YES	
asin	YES	
asinh	YES	
atan	YES	
atan2	YES	
atanh	YES	
cbrt	YES	
ceil	YES	

Function	Migration Support	Diagnostic Message
copysign	YES	
cos	YES	
cosh	YES	
cospi	YES	
cyl_bessel_i0	YES	
cyl_bessel_i1	YES	
erf	YES	
erfc	YES	
erfcinv	YES	
erfcx	YES	DPCT1017
erfinv	YES	
exp	YES	
exp10	YES	
exp2	YES	
expm1	YES	
fabs	YES	
fdim	YES	
floor	YES	
fma	YES	
fmax	YES	
fmin	YES	
fmod	YES	
frexp	YES	DPCT1017
hypot	YES	
ilogb	YES	
j0	YES	
j1	YES	
jn	YES	
ldexp	YES	
lgamma	YES	

Function	Migration Support	Diagnostic Message
llrint	YES	
llround	YES	
log	YES	
log10	YES	
log1p	YES	
log2	YES	
logb	YES	
lrint	YES	
lround	YES	
modf	YES	DPCT1017
nan	YES	DPCT1017
nearbyint	YES	DPCT1017
nextafter	YES	
norm	YES	DPCT1017
norm3d	YES	
norm4d	YES	
normcdf	YES	
normcdfinv	YES	
pow	YES	
rcbrt	YES	DPCT1017
remainder	YES	
remquo	YES	DPCT1017
rhypot	YES	DPCT1017
rint	YES	
rnorm	YES	DPCT1017
rnorm3d	YES	
rnorm4d	YES	
round	YES	
rsqrt	YES	
scalbln	YES	DPCT1017

Function	Migration Support	Diagnostic Message
scalbn	YES	DPCT1017
sin	YES	
sincos	YES	DPCT1017
sincospi	YES	DPCT1017
sinh	YES	
sinpi	YES	
sqrt	YES	
tan	YES	
tanh	YES	
tgamma	YES	
trunc	YES	
y0	YES	
y1	YES	
yn	YES	
abs	YES	
labs	YES	
llabs	YES	
llmax	YES	
llmin	YES	
ullmax	YES	
ullmin	YES	
umax	YES	
umin	YES	
__cosf	YES	
__exp10f	YES	
__expf	YES	
__fadd_rd	YES	DPCT1013
__fadd_rn	YES	DPCT1013
__fadd_ru	YES	DPCT1013
__fadd_rz	YES	DPCT1013

Function	Migration Support	Diagnostic Message
__fdiv_rd	YES	DPCT1013
__fdiv_rn	YES	DPCT1013
__fdiv_ru	YES	DPCT1013
__fdiv_rz	YES	DPCT1013
__fdividef	YES	
__fmaf_ieee_rd	NO	
__fmaf_ieee_rn	NO	
__fmaf_ieee_ru	NO	
__fmaf_ieee_rz	NO	
__fmaf_rd	YES	DPCT1013
__fmaf_rn	YES	DPCT1013
__fmaf_ru	YES	DPCT1013
__fmaf_rz	YES	DPCT1013
__fmul_rd	YES	DPCT1013
__fmul_rn	YES	DPCT1013
__fmul_ru	YES	DPCT1013
__fmul_rz	YES	DPCT1013
__frcp_rd	YES	DPCT1013
__frcp_rn	YES	DPCT1013
__frcp_ru	YES	DPCT1013
__frcp_rz	YES	DPCT1013
__frsqrt_rn	YES	DPCT1013
__fsqrt_rd	YES	DPCT1013
__fsqrt_rn	YES	DPCT1013
__fsqrt_ru	YES	DPCT1013
__fsqrt_rz	YES	DPCT1013
__fsub_rd	YES	DPCT1013
__fsub_rn	YES	DPCT1013
__fsub_ru	YES	DPCT1013
__fsub_rz	YES	DPCT1013

Function	Migration Support	Diagnostic Message
__log10f	YES	
__log2f	YES	
__logf	YES	
__powf	YES	
__saturatef	YES	
__sincosf	YES	DPCT1017
__sinf	YES	
__tanf	YES	
__dadd_rd	YES	DPCT1013
__dadd_rn	YES	DPCT1013
__dadd_ru	YES	DPCT1013
__dadd_rz	YES	DPCT1013
__ddiv_rd	YES	DPCT1013
__ddiv_rn	YES	DPCT1013
__ddiv_ru	YES	DPCT1013
__ddiv_rz	YES	DPCT1013
__dmul_rd	YES	DPCT1013
__dmul_rn	YES	DPCT1013
__dmul_ru	YES	DPCT1013
__dmul_rz	YES	DPCT1013
__drcp_rd	YES	DPCT1013
__drcp_rn	YES	DPCT1013
__drcp_ru	YES	DPCT1013
__drcp_rz	YES	DPCT1013
__dsqrt_rd	YES	DPCT1013
__dsqrt_rn	YES	DPCT1013
__dsqrt_ru	YES	DPCT1013
__dsqrt_rz	YES	DPCT1013
__dsub_rd	YES	DPCT1013
__dsub_rn	YES	DPCT1013

Function	Migration Support	Diagnostic Message
__dsub_ru	YES	DPCT1013
__dsub_rz	YES	DPCT1013
__fma_rd	YES	DPCT1013
__fma_rn	YES	DPCT1013
__fma_ru	YES	DPCT1013
__fma_rz	YES	DPCT1013
__brev	YES	
__brevll	YES	
__byte_perm	YES	
__clz	YES	
__clzll	YES	
__dp2a_hi	YES	
__dp2a_lo	YES	
__dp4a	YES	
__ffs	YES	
__ffsll	YES	
__fns	NO	
__funnelshift_l	YES	DPCT1098
__funnelshift_lc	YES	DPCT1098
__funnelshift_r	YES	DPCT1098
__funnelshift_rc	YES	DPCT1098
__mul24	YES	
__mul64hi	YES	
__mulhi	YES	
__popc	YES	
__popcll	YES	
__rhadd	YES	
__sad	YES	DPCT1017
__uhadd	YES	
__umul24	YES	

Function	Migration Support	Diagnostic Message
__umul64hi	YES	
__umulhi	YES	
__urhadd	YES	
__usad	YES	DPCT1017
__double2float_rd	YES	
__double2float_rn	YES	
__double2float_ru	YES	
__double2float_rz	YES	
__double2hiint	YES	DPCT1017
__double2int_rd	YES	
__double2int_rn	YES	
__double2int_ru	YES	
__double2int_rz	YES	
__double2ll_rd	YES	
__double2ll_rn	YES	
__double2ll_ru	YES	
__double2ll_rz	YES	
__double2loint	YES	DPCT1017
__double2uint_rd	YES	
__double2uint_rn	YES	
__double2uint_ru	YES	
__double2uint_rz	YES	
__double2ull_rd	YES	
__double2ull_rn	YES	
__double2ull_ru	YES	
__double2ull_rz	YES	
__double_as_longlong	YES	
__float2int_rd	YES	
__float2int_rn	YES	
__float2int_ru	YES	

Function	Migration Support	Diagnostic Message
__float2int_rz	YES	
__float2ll_rd	YES	
__float2ll_rn	YES	
__float2ll_ru	YES	
__float2ll_rz	YES	
__float2uint_rd	YES	
__float2uint_rn	YES	
__float2uint_ru	YES	
__float2uint_rz	YES	
__float2ull_rd	YES	
__float2ull_rn	YES	
__float2ull_ru	YES	
__float2ull_rz	YES	
__float_as_int	YES	
__float_as_uint	YES	
__hiloint2double	YES	DPCT1017
__int2double_rn	YES	
__int2float_rd	YES	
__int2float_rn	YES	
__int2float_ru	YES	
__int2float_rz	YES	
__int_as_float	YES	
__ll2double_rd	YES	
__ll2double_rn	YES	
__ll2double_ru	YES	
__ll2double_rz	YES	
__ll2float_rd	YES	
__ll2float_rn	YES	
__ll2float_ru	YES	
__ll2float_rz	YES	

Function	Migration Support	Diagnostic Message
__longlong_as_double	YES	
__uint2double_rn	YES	
__uint2float_rd	YES	
__uint2float_rn	YES	
__uint2float_ru	YES	
__uint2float_rz	YES	
__uint_as_float	YES	
__ull2double_rd	YES	
__ull2double_rn	YES	
__ull2double_ru	YES	
__ull2double_rz	YES	
__ull2float_rd	YES	
__ull2float_rn	YES	
__ull2float_ru	YES	
__ull2float_rz	YES	
__vabs2	YES	
__vabs4	YES	
__vabsdiffs2	YES	
__vabsdiffs4	YES	
__vabsdiffu2	YES	
__vabsdiffu4	YES	
__vabsss2	YES	
__vabsss4	YES	
__vadd2	YES	
__vadd4	YES	
__vaddss2	YES	
__vaddss4	YES	
__vaddus2	YES	
__vaddus4	YES	
__vavgs2	YES	

Function	Migration Support	Diagnostic Message
__vavgs4	YES	
__vavgu2	YES	
__vavgu4	YES	
__vcmpeq2	YES	
__vcmpeq4	YES	
__vcmpges2	YES	
__vcmpges4	YES	
__vcmpgeu2	YES	
__vcmpgeu4	YES	
__vcmpgts2	YES	
__vcmpgts4	YES	
__vcmpgtu2	YES	
__vcmpgtu4	YES	
__vcmples2	YES	
__vcmples4	YES	
__vcmpleu2	YES	
__vcmpleu4	YES	
__vcmplts2	YES	
__vcmplts4	YES	
__vcmpltu2	YES	
__vcmpltu4	YES	
__vcmpne2	YES	
__vcmpne4	YES	
__vhaddu2	YES	
__vhaddu4	YES	
__viaddmax_s16x2	NO	
__viaddmax_s16x2_relu	NO	
__viaddmax_s32	NO	
__viaddmax_s32_relu	NO	
__viaddmax_u16x2	NO	

Function	Migration Support	Diagnostic Message
__viaddmax_u32	NO	
__viaddmin_s16x2	NO	
__viaddmin_s16x2_relu	NO	
__viaddmin_s32	NO	
__viaddmin_s32_relu	NO	
__viaddmin_u16x2	NO	
__viaddmin_u32	NO	
__vibmax_s16x2	NO	
__vibmax_s32	NO	
__vibmax_u16x2	NO	
__vibmax_u32	NO	
__vibmin_s16x2	NO	
__vibmin_s32	NO	
__vibmin_u16x2	NO	
__vibmin_u32	NO	
__vimax3_s16x2	NO	
__vimax3_s16x2_relu	NO	
__vimax3_s32	NO	
__vimax3_s32_relu	NO	
__vimax3_u16x2	NO	
__vimax3_u32	NO	
__vimax_s16x2_relu	NO	
__vimax_s32_relu	NO	
__vimin3_s16x2	NO	
__vimin3_s16x2_relu	NO	
__vimin3_s32	NO	
__vimin3_s32_relu	NO	
__vimin3_u16x2	NO	
__vimin3_u32	NO	
__vimin_s16x2_relu	NO	

Function	Migration Support	Diagnostic Message
__vimin_s32_relu	NO	
__vmaxs2	YES	
__vmaxs4	YES	
__vmaxu2	YES	
__vmaxu4	YES	
__vmins2	YES	
__vmins4	YES	
__vminu2	YES	
__vminu4	YES	
__vneg2	YES	
__vneg4	YES	
__vnegss2	YES	
__vnegss4	YES	
__vsads2	YES	
__vsads4	YES	
__vsadu2	YES	
__vsadu4	YES	
__vseteq2	YES	
__vseteq4	YES	
__vsetges2	YES	
__vsetges4	YES	
__vsetgeu2	YES	
__vsetgeu4	YES	
__vsetgts2	YES	
__vsetgts4	YES	
__vsetgtu2	YES	
__vsetgtu4	YES	
__vsetles2	YES	
__vsetles4	YES	
__vsetleu2	YES	

Function	Migration Support	Diagnostic Message
__vsetleu4	YES	
__vsetlts2	YES	
__vsetlts4	YES	
__vsetltu2	YES	
__vsetltu4	YES	
__vsetne2	YES	
__vsetne4	YES	
__vsub2	YES	
__vsub4	YES	
__vsubss2	YES	
__vsubss4	YES	
__vsubus2	YES	
__vsubus4	YES	
acospi	NO	
tanpi	NO	
__cudaGetExportTableInternal	NO	
__cudaInitManagedRuntime	NO	
__cudaInitModule	NO	
__cudaRegisterDeviceFunction	NO	
__cudaRegisterFatBinary	NO	
__cudaRegisterFunction	NO	
__cudaRegisterManagedVar	NO	
__cudaRegisterPrelinkedFatBinary	NO	
__cudaRegisterShared	NO	
__cudaRegisterSharedVar	NO	
__cudaRegisterSurface	NO	
__cudaRegisterTexture	NO	
__cudaRegisterVar	NO	
__cudaUnregisterFatBinary	NO	
h2div	YES	

Function	Migration Support	Diagnostic Message
hdiv	YES	
make_char1	YES	
make_char2	YES	
make_char3	YES	
make_char4	YES	
make_double1	YES	
make_double2	YES	
make_double3	YES	
make_double4	YES	
make_float1	YES	
make_float2	YES	
make_float3	YES	
make_float4	YES	
make_int1	YES	
make_int2	YES	
make_int3	YES	
make_int4	YES	
make_long1	YES	
make_long2	YES	
make_long3	YES	
make_long4	YES	
make_longlong1	YES	
make_longlong2	YES	
make_longlong3	YES	
make_longlong4	YES	
make_short1	YES	
make_short2	YES	
make_short3	YES	
make_short4	YES	
make_uchar1	YES	

Function	Migration Support	Diagnostic Message
make_uchar2	YES	
make_uchar3	YES	
make_uchar4	YES	
make_uint1	YES	
make_uint2	YES	
make_uint3	YES	
make_uint4	YES	
make_ulong1	YES	
make_ulong2	YES	
make_ulong3	YES	
make_ulong4	YES	
make_ulonglong1	YES	
make_ulonglong2	YES	
make_ulonglong3	YES	
make_ulonglong4	YES	
make_ushort1	YES	
make_ushort2	YES	
make_ushort3	YES	
make_ushort4	YES	
__threadfence_block	YES	DPCT1078
__threadfence	YES	DPCT1078
__threadfence_system	YES	DPCT1078
__barrier_sync	YES	DPCT1078
__syncthreads	YES	DPCT1078
__syncthreads_count	YES	DPCT1078
__syncthreads_and	YES	DPCT1078
__syncthreads_or	YES	DPCT1078
tex1Dfetch	YES	
tex1D	YES	
tex1DLod	YES	DPCT1119

Function	Migration Support	Diagnostic Message
tex1DGrad	NO	
tex2D	YES	
tex2DLod	YES	DPCT1119
tex2DGrad	NO	
tex3D	YES	
tex3DLod	YES	DPCT1119
tex3DGrad	NO	
tex1DLayered	YES	
tex1DLayeredLod	NO	
tex1DLayeredGrad	NO	
tex2DLayered	YES	
tex2DLayeredLod	NO	
tex2DLayeredGrad	NO	
texCubemap	NO	
texCubemapLod	NO	
texCubemapLayered	NO	
texCubemapLayeredLod	NO	
tex2Dgather	NO	
surf1Dread	NO	
surf1Dwrite	NO	
surf2Dread	NO	
surf2Dwrite	NO	
surf3Dread	NO	
surf3Dwrite	NO	
surf1DLayeredread	NO	
surf1DLayeredwrite	NO	
surf2DLayeredread	NO	
surf2DLayeredwrite	NO	
surfCubemapread	NO	
surfCubemapwrite	NO	

Function	Migration Support	Diagnostic Message
surfCubemapLayeredread	NO	
surfCubemapLayeredwrite	NO	
clock	YES	DPCT1008
clock64	YES	DPCT1008
__nanosleep	YES	DPCT1008
atomicAdd_system	YES	
atomicSub_system	YES	
atomicExch_system	YES	
atomicMin_system	YES	
atomicMax_system	YES	
atomicInc_system	YES	
atomicDec_system	YES	
atomicCAS_system	YES	
atomicAnd_system	YES	
atomicOr_system	YES	
atomicXor_system	YES	
__all	YES	
__any	YES	
__ballot	YES	
__shfl	YES	
__shfl_up	YES	
__shfl_down	YES	
__shfl_xor	YES	
__prof_trigger	NO	
asm	NO	
__syncwarp	YES	
__all_sync	YES	
__any_sync	YES	
__ballot_sync	YES	
__reduce_add_sync	YES	

Function	Migration Support	Diagnostic Message
__reduce_min_sync	YES	
__reduce_max_sync	YES	
__reduce_and_sync	YES	
__reduce_or_sync	YES	
__reduce_xor_sync	YES	
__activemask	YES	DPCT1086
__match_any_sync	YES	
__match_all_sync	YES	
cuGetErrorName	NO	
cuGetErrorString	YES	
cuInit	YES	DPCT1026 / DPCT1027
cuDriverGetVersion	YES	DPCT1043
cuDeviceGet	YES	
cuDeviceGetAttribute	YES	DPCT1051 / DPCT1076
cuDeviceGetCount	YES	
cuDeviceGetDefaultMemPool	NO	
cuDeviceGetExecAffinitySupport	NO	
cuDeviceGetLuid	NO	
cuDeviceGetMemPool	NO	
cuDeviceGetName	YES	
cuDeviceGetNvSciSyncAttributes	NO	
cuDeviceGetTexture1DLinearMaxWidth	NO	
cuDeviceGetUuid	YES	DPCT1090
cuDeviceGetUuid_v2	YES	DPCT1090
cuDeviceSetMemPool	NO	
cuDeviceTotalMem	YES	
cuFlushGPUDirectRDMAWrites	NO	
cuDeviceComputeCapability	YES	
cuDeviceGetProperties	NO	
cuDevicePrimaryCtxGetState	NO	

Function	Migration Support	Diagnostic Message
cuDevicePrimaryCtxRelease	YES	DPCT1026 / DPCT1027
cuDevicePrimaryCtxReset	NO	
cuDevicePrimaryCtxRetain	YES	
cuDevicePrimaryCtxSetFlags	NO	
cuCtxCreate	YES	
cuCtxCreate_v3	YES	
cuCtxCreate_v4	YES	
cuCtxDestroy	YES	DPCT1026 / DPCT1027
cuCtxGetApiVersion	YES	
cuCtxGetCacheConfig	NO	
cuCtxGetCurrent	YES	
cuCtxGetDevice	YES	
cuCtxGetExecAffinity	NO	
cuCtxGetFlags	NO	
cuCtxGetId	NO	
cuCtxGetLimit	YES	DPCT1026 / DPCT1027
cuCtxGetStreamPriorityRange	NO	
cuCtxPopCurrent	YES	
cuCtxPushCurrent	YES	
cuCtxRecordEvent	NO	
cuCtxResetPersistingL2Cache	YES	DPCT1026 / DPCT1027
cuCtxSetCacheConfig	YES	DPCT1026 / DPCT1027
cuCtxSetCurrent	YES	
cuCtxSetFlags	NO	
cuCtxSetLimit	YES	DPCT1026 / DPCT1027
cuCtxSynchronize	YES	
cuCtxWaitEvent	NO	
cuCtxAttach	NO	
cuCtxDetach	NO	
cuCtxGetSharedMemConfig	NO	

Function	Migration Support	Diagnostic Message
cuCtxSetSharedMemConfig	NO	
cuLinkAddData	NO	
cuLinkAddFile	NO	
cuLinkComplete	NO	
cuLinkCreate	NO	
cuLinkDestroy	NO	
cuModuleEnumerateFunctions	NO	
cuModuleGetFunction	YES	
cuModuleGetFunctionCount	NO	
cuModuleGetGlobal	NO	
cuModuleGetLoadingMode	NO	
cuModuleLoad	YES	DPCT1103
cuModuleLoadData	YES	DPCT1104
cuModuleLoadDataEx	YES	DPCT1104
cuModuleLoadFatBinary	NO	
cuModuleUnload	YES	
cuModuleGetSurfRef	NO	
cuModuleGetTexRef	YES	
cuKernelGetAttribute	NO	
cuKernelGetFunction	NO	
cuKernelGetLibrary	NO	
cuKernelGetName	NO	
cuKernelGetParamInfo	NO	
cuKernelSetAttribute	NO	
cuKernelSetCacheConfig	NO	
cuLibraryEnumerateKernels	NO	
cuLibraryGetGlobal	NO	
cuLibraryGetKernel	NO	
cuLibraryGetKernelCount	NO	
cuLibraryGetManaged	NO	

Function	Migration Support	Diagnostic Message
cuLibraryGetModule	NO	
cuLibraryGetUnifiedFunction	NO	
cuLibraryLoadData	NO	
cuLibraryLoadFromFile	NO	
cuLibraryUnload	NO	
cuArray3DCreate	YES	
cuArray3DGetDescriptor	NO	
cuArrayCreate	YES	
cuArrayDestroy	YES	
cuArrayGetDescriptor	NO	
cuArrayGetMemoryRequirements	NO	
cuArrayGetPlane	NO	
cuArrayGetSparseProperties	NO	
cuDeviceGetByPCIBusId	NO	
cuDeviceGetPCIBusId	NO	
cuDeviceRegisterAsyncNotification	NO	
cuDeviceUnregisterAsyncNotification	NO	
cuIpcCloseMemHandle	NO	
cuIpcGetEventHandle	NO	
cuIpcGetMemHandle	NO	
cuIpcOpenEventHandle	NO	
cuIpcOpenMemHandle	NO	
cuMemAlloc	YES	
cuMemAllocHost	YES	
cuMemAllocManaged	YES	
cuMemAllocPitch	YES	
cuMemFree	YES	
cuMemFreeHost	YES	
cuMemGetAddressRange	NO	
cuMemGetHandleForAddressRange	NO	

Function	Migration Support	Diagnostic Message
cuMemGetInfo	YES	DPCT1072 / DPCT1106
cuMemHostAlloc	YES	DPCT1072
cuMemHostGetDevicePointer	YES	
cuMemHostGetFlags	YES	
cuMemHostRegister	YES	DPCT1026 / DPCT1027
cuMemHostUnregister	YES	DPCT1026 / DPCT1027
cuMemcpy	YES	
cuMemcpy2D	YES	
cuMemcpy2DAsync	YES	
cuMemcpy2DUnaligned	NO	
cuMemcpy3D	YES	
cuMemcpy3DAsync	YES	
cuMemcpy3DPeer	YES	
cuMemcpy3DPeerAsync	YES	
cuMemcpyAsync	YES	
cuMemcpyAtoA	YES	
cuMemcpyAtoD	YES	
cuMemcpyAtoH	YES	
cuMemcpyAtoHAsync	YES	
cuMemcpyDtoA	YES	
cuMemcpyDtoD	YES	
cuMemcpyDtoDAsync	YES	
cuMemcpyDtoH	YES	
cuMemcpyDtoHAsync	YES	
cuMemcpyHtoA	YES	
cuMemcpyHtoAAsync	YES	
cuMemcpyHtoD	YES	
cuMemcpyHtoDAsync	YES	
cuMemcpyPeer	YES	
cuMemcpyPeerAsync	YES	

Function	Migration Support	Diagnostic Message
cuMemsetD16	YES	
cuMemsetD16Async	YES	
cuMemsetD2D16	YES	
cuMemsetD2D16Async	YES	
cuMemsetD2D32	YES	
cuMemsetD2D32Async	YES	
cuMemsetD2D8	YES	
cuMemsetD2D8Async	YES	
cuMemsetD32	YES	
cuMemsetD32Async	YES	
cuMemsetD8	YES	
cuMemsetD8Async	YES	
cuMipmappedArrayCreate	NO	
cuMipmappedArrayDestroy	NO	
cuMipmappedArrayGetLevel	NO	
cuMipmappedArrayGetMemoryRequirements	NO	
cuMipmappedArrayGetSparseProperties	NO	
cuMemAddressFree	NO	
cuMemAddressReserve	NO	
cuMemCreate	NO	
cuMemExportToShareableHandle	NO	
cuMemGetAccess	NO	
cuMemGetAllocationGranularity	NO	
cuMemGetAllocationPropertiesFromHandle	NO	
cuMemImportFromShareableHandle	NO	
cuMemMap	NO	
cuMemMapArrayAsync	NO	
cuMemRelease	NO	
cuMemRetainAllocationHandle	NO	

Function	Migration Support	Diagnostic Message
cuMemSetAccess	NO	
cuMemUnmap	NO	
cuMemAllocAsync	NO	
cuMemAllocFromPoolAsync	NO	
cuMemFreeAsync	NO	
cuMemPoolCreate	NO	
cuMemPoolDestroy	NO	
cuMemPoolExportPointer	NO	
cuMemPoolExportToShareableHandle	NO	
cuMemPoolGetAccess	NO	
cuMemPoolGetAttribute	NO	
cuMemPoolImportFromShareableHandle	NO	
cuMemPoolImportPointer	NO	
cuMemPoolSetAccess	NO	
cuMemPoolSetAttribute	NO	
cuMemPoolTrimTo	NO	
cuMulticastAddDevice	NO	
cuMulticastBindAddr	NO	
cuMulticastBindMem	NO	
cuMulticastCreate	NO	
cuMulticastGetGranularity	NO	
cuMulticastUnbind	NO	
cuMemAdvise	YES	DPCT1063 Partial
cuMemAdvise_v2	NO	Partial
cuMemPrefetchAsync	YES	Partial
cuMemPrefetchAsync_v2	NO	Partial
cuMemRangeGetAttribute	NO	
cuMemRangeGetAttributes	NO	
cuPointerGetAttribute	NO	
cuPointerGetAttributes	YES	Partial

Function	Migration Support	Diagnostic Message
cuPointerSetAttribute	NO	
cuStreamAddCallback	YES	
cuStreamAttachMemAsync	YES	DPCT1026 / DPCT1027
cuStreamBeginCapture	NO	
cuStreamBeginCapture_ptsz	NO	
cuStreamBeginCaptureToGraph	NO	
cuStreamCopyAttributes	NO	
cuStreamCreate	YES	
cuStreamCreateWithPriority	NO	
cuStreamDestroy	YES	
cuStreamEndCapture	NO	
cuStreamGetAttribute	NO	
cuStreamGetCaptureInfo	NO	
cuStreamGetCaptureInfo_v3	NO	
cuStreamGetCtx	NO	
cuStreamGetCtx_v2	NO	
cuStreamGetFlags	NO	
cuStreamGetId	NO	
cuStreamGetPriority	NO	
cuStreamIsCapturing	NO	
cuStreamQuery	YES	DPCT1026 / DPCT1027
cuStreamSetAttribute	NO	
cuStreamSynchronize	YES	
cuStreamUpdateCaptureDependencies	NO	
cuStreamUpdateCaptureDependencies_v2	NO	
cuStreamWaitEvent	YES	
cuThreadExchangeStreamCaptureMode	NO	
cuEventCreate	YES	DPCT1026 / DPCT1027
cuEventDestroy	YES	DPCT1026 / DPCT1027
cuEventElapsedTime	YES	DPCT1012

Function	Migration Support	Diagnostic Message
cuEventQuery	YES	
cuEventRecord	YES	DPCT1012 / DPCT1024
cuEventRecordWithFlags	NO	
cuEventSynchronize	YES	DPCT1012
cuDestroyExternalMemory	NO	
cuDestroyExternalSemaphore	NO	
cuExternalMemoryGetMappedBuffer	NO	
cuExternalMemoryGetMappedMipmappedArray	NO	
cuImportExternalMemory	NO	
cuImportExternalSemaphore	NO	
cuSignalExternalSemaphoresAsync	NO	
cuWaitExternalSemaphoresAsync	NO	
cuStreamBatchMemOp	NO	
cuStreamWaitValue32	NO	
cuStreamWaitValue64	NO	
cuStreamWriteValue32	NO	
cuStreamWriteValue64	NO	
cuFuncGetAttribute	YES	
cuFuncGetModule	NO	
cuFuncGetName	NO	
cuFuncGetParamInfo	NO	
cuFuncIsLoaded	NO	
cuFuncLoad	NO	
cuFuncSetAttribute	YES	DPCT1026
cuFuncSetCacheConfig	YES	DPCT1026
cuLaunchCooperativeKernel	NO	
cuLaunchCooperativeKernelMultiDevice	NO	
cuLaunchHostFunc	NO	
cuLaunchKernel	YES	
cuLaunchKernelEx	NO	

Function	Migration Support	Diagnostic Message
cuFuncSetBlockShape	NO	
cuFuncSetSharedMemConfig	NO	
cuFuncSetSharedSize	NO	
cuLaunch	NO	
cuLaunchGrid	NO	
cuLaunchGridAsync	NO	
cuParamSetSize	NO	
cuParamSetTexRef	NO	
cuParamSetf	NO	
cuParamSeti	NO	
cuParamSetv	NO	
cuDeviceGetGraphMemAttribute	NO	
cuDeviceGraphMemTrim	NO	
cuDeviceSetGraphMemAttribute	NO	
cuGraphAddBatchMemOpNode	NO	
cuGraphAddChildGraphNode	NO	
cuGraphAddDependencies	NO	
cuGraphAddDependencies_v2	NO	
cuGraphAddEmptyNode	NO	
cuGraphAddEventRecordNode	NO	
cuGraphAddEventWaitNode	NO	
cuGraphAddExternalSemaphoresSignalNode	NO	
cuGraphAddExternalSemaphoresWaitNode	NO	
cuGraphAddHostNode	NO	
cuGraphAddKernelNode	NO	
cuGraphAddMemAllocNode	NO	
cuGraphAddMemFreeNode	NO	
cuGraphAddMemcpyNode	NO	
cuGraphAddMemsetNode	NO	

Function	Migration Support	Diagnostic Message
cuGraphAddNode	NO	
cuGraphAddNode_v2	NO	
cuGraphBatchMemOpNodeGetParams	NO	
cuGraphBatchMemOpNodeSetParams	NO	
cuGraphChildGraphNodeGetGraph	NO	
cuGraphClone	NO	
cuGraphConditionalHandleCreate	NO	
cuGraphCreate	NO	
cuGraphDebugDotPrint	NO	
cuGraphDestroy	NO	
cuGraphDestroyNode	NO	
cuGraphEventRecordNodeGetEvent	NO	
cuGraphEventRecordNodeSetEvent	NO	
cuGraphEventWaitNodeGetEvent	NO	
cuGraphEventWaitNodeSetEvent	NO	
cuGraphExecBatchMemOpNodeSetParams	NO	
cuGraphExecChildGraphNodeSetParams	NO	
cuGraphExecDestroy	NO	
cuGraphExecEventRecordNodeSetEvent	NO	
cuGraphExecEventWaitNodeSetEvent	NO	
cuGraphExecExternalSemaphoresSignalNodeSetParams	NO	
cuGraphExecExternalSemaphoresWaitNodeSetParams	NO	
cuGraphExecGetFlags	NO	
cuGraphExecHostNodeSetParams	NO	
cuGraphExecKernelNodeSetParams	NO	
cuGraphExecMemcpyNodeSetParams	NO	
cuGraphExecMemsetNodeSetParams	NO	
cuGraphExecNodeSetParams	NO	
cuGraphExecUpdate	NO	

Function	Migration Support	Diagnostic Message
cuGraphExternalSemaphoresSignalNodeGetParams	NO	
cuGraphExternalSemaphoresSignalNodeSetParams	NO	
cuGraphExternalSemaphoresWaitNodeGetParams	NO	
cuGraphExternalSemaphoresWaitNodeSetParams	NO	
cuGraphGetEdges	NO	
cuGraphGetEdges_v2	NO	
cuGraphGetNodes	NO	
cuGraphGetRootNodes	NO	
cuGraphHostNodeGetParams	NO	
cuGraphHostNodeSetParams	NO	
cuGraphInstantiate	NO	
cuGraphInstantiateWithParams	NO	
cuGraphKernelNodeCopyAttributes	NO	
cuGraphKernelNodeGetAttribute	NO	
cuGraphKernelNodeGetParams	NO	
cuGraphKernelNodeSetAttribute	NO	
cuGraphKernelNodeSetParams	NO	
cuGraphLaunch	NO	
cuGraphMemAllocNodeGetParams	NO	
cuGraphMemFreeNodeGetParams	NO	
cuGraphMemcpyNodeGetParams	NO	
cuGraphMemcpyNodeSetParams	NO	
cuGraphMemsetNodeGetParams	NO	
cuGraphMemsetNodeSetParams	NO	
cuGraphNodeFindInClone	NO	
cuGraphNodeGetDependencies	NO	
cuGraphNodeGetDependencies_v2	NO	
cuGraphNodeGetDependentNodes	NO	

Function	Migration Support	Diagnostic Message
cuGraphNodeGetDependentNodes_v2	NO	
cuGraphNodeGetEnabled	NO	
cuGraphNodeGetType	NO	
cuGraphNodeSetEnabled	NO	
cuGraphNodeSetParams	NO	
cuGraphReleaseUserObject	NO	
cuGraphRemoveDependencies	NO	
cuGraphRemoveDependencies_v2	NO	
cuGraphRetainUserObject	NO	
cuGraphUpload	NO	
cuUserObjectCreate	NO	
cuUserObjectRelease	NO	
cuUserObjectRetain	NO	
cuGraphInstantiateWithFlags	NO	
cuOccupancyAvailableDynamicSMemPerBlock	NO	
cuOccupancyMaxActiveBlocksPerMultiprocessor	YES	DPCT1111
cuOccupancyMaxActiveBlocksPerMultiprocessorWithFlags	NO	
cuOccupancyMaxActiveClusters	NO	
cuOccupancyMaxPotentialBlockSize	NO	
cuOccupancyMaxPotentialBlockSizeWithFlags	NO	
cuOccupancyMaxPotentialClusterSize	NO	
cuTexRefCreate	NO	
cuTexRefDestroy	NO	
cuTexRefGetAddress	NO	
cuTexRefGetAddressMode	YES	
cuTexRefGetArray	NO	
cuTexRefGetBorderColor	NO	
cuTexRefGetFilterMode	YES	

Function	Migration Support	Diagnostic Message
cuTexRefGetFlags	YES	
cuTexRefGetFormat	NO	
cuTexRefGetMaxAnisotropy	NO	
cuTexRefGetMipmapFilterMode	NO	
cuTexRefGetMipmapLevelBias	NO	
cuTexRefGetMipmapLevelClamp	NO	
cuTexRefGetMipmappedArray	NO	
cuTexRefSetAddress	YES	
cuTexRefSetAddress2D	YES	
cuTexRefSetAddressMode	YES	
cuTexRefSetArray	YES	
cuTexRefSetBorderColor	NO	
cuTexRefSetFilterMode	YES	
cuTexRefSetFlags	YES	DPCT1074
cuTexRefSetFormat	YES	
cuTexRefSetMaxAnisotropy	NO	
cuTexRefSetMipmapFilterMode	NO	
cuTexRefSetMipmapLevelBias	NO	
cuTexRefSetMipmapLevelClamp	NO	
cuTexRefSetMipmappedArray	NO	
cuSurfRefGetArray	NO	
cuSurfRefSetArray	NO	
cuTexObjectCreate	YES	
cuTexObjectDestroy	YES	
cuTexObjectGetResourceDesc	YES	
cuTexObjectGetResourceViewDesc	NO	
cuTexObjectGetTextureDesc	YES	
cuSurfObjectCreate	NO	
cuSurfObjectDestroy	NO	
cuSurfObjectGetResourceDesc	NO	

Function	Migration Support	Diagnostic Message
cuTensorMapEncodeIm2col	NO	
cuTensorMapEncodeTiled	NO	
cuTensorMapReplaceAddress	NO	
cuCtxDisablePeerAccess	NO	
cuCtxEnablePeerAccess	YES	DPCT1026 / DPCT1027
cuDeviceCanAccessPeer	YES	DPCT1031
cuDeviceGetP2PAttribute	NO	
cuGraphicsMapResources	NO	
cuGraphicsResourceGetMappedMipmappedArray	NO	
cuGraphicsResourceGetMappedPointer	NO	
cuGraphicsResourceSetMapFlags	NO	
cuGraphicsSubResourceGetMappedArray	NO	
cuGraphicsUnmapResources	NO	
cuGraphicsUnregisterResource	NO	
cuGetProcAddress	NO	
cuCoredumpGetAttribute	NO	
cuCoredumpGetAttributeGlobal	NO	
cuCoredumpSetAttribute	NO	
cuCoredumpSetAttributeGlobal	NO	
cuCtxFromGreenCtx	NO	
cuCtxGetDevResource	NO	
cuDevResourceGenerateDesc	NO	
cuDevSmResourceSplitByCount	NO	
cuDeviceGetDevResource	NO	
cuGreenCtxCreate	NO	
cuGreenCtxDestroy	NO	
cuGreenCtxGetDevResource	NO	
cuGreenCtxRecordEvent	NO	
cuGreenCtxStreamCreate	NO	
cuGreenCtxWaitEvent	NO	

Function	Migration Support	Diagnostic Message
cuStreamGetGreenCtx	NO	
cuProfilerInitialize	NO	
cuProfilerStart	NO	
cuProfilerStop	NO	
cuGLGetDevices	NO	
cuGraphicsGLRegisterBuffer	NO	
cuGraphicsGLRegisterImage	NO	
cuWGLGetDevice	NO	
cuGLCtxCreate	NO	
cuGLInit	NO	
cuGLMapBufferObject	NO	
cuGLMapBufferObjectAsync	NO	
cuGLRegisterBufferObject	NO	
cuGLSetBufferObjectMapFlags	NO	
cuGLUnmapBufferObject	NO	
cuGLUnmapBufferObjectAsync	NO	
cuGLUnregisterBufferObject	NO	
cuD3D9CtxCreate	NO	
cuD3D9CtxCreateOnDevice	NO	
cuD3D9GetDevice	NO	
cuD3D9GetDevices	NO	
cuD3D9GetDirect3DDevice	NO	
cuGraphicsD3D9RegisterResource	NO	
cuD3D9MapResources	NO	
cuD3D9RegisterResource	NO	
cuD3D9ResourceGetMappedArray	NO	
cuD3D9ResourceGetMappedPitch	NO	
cuD3D9ResourceGetMappedPointer	NO	
cuD3D9ResourceGetMappedSize	NO	
cuD3D9ResourceGetSurfaceDimensions	NO	

Function	Migration Support	Diagnostic Message
cuD3D9ResourceSetMapFlags	NO	
cuD3D9UnmapResources	NO	
cuD3D9UnregisterResource	NO	
cuD3D10GetDevice	NO	
cuD3D10GetDevices	NO	
cuGraphicsD3D10RegisterResource	NO	
cuD3D10CtxCreate	NO	
cuD3D10CtxCreateOnDevice	NO	
cuD3D10GetDirect3DDevice	NO	
cuD3D10MapResources	NO	
cuD3D10RegisterResource	NO	
cuD3D10ResourceGetMappedArray	NO	
cuD3D10ResourceGetMappedPitch	NO	
cuD3D10ResourceGetMappedPointer	NO	
cuD3D10ResourceGetMappedSize	NO	
cuD3D10ResourceGetSurfaceDimensions	NO	
cuD3D10ResourceSetMapFlags	NO	
cuD3D10UnmapResources	NO	
cuD3D10UnregisterResource	NO	
cuD3D11GetDevice	NO	
cuD3D11GetDevices	NO	
cuGraphicsD3D11RegisterResource	NO	
cuD3D11CtxCreate	NO	
cuD3D11CtxCreateOnDevice	NO	
cuD3D11GetDirect3DDevice	NO	
cuGraphicsVDPAURegisterOutputSurface	NO	
cuGraphicsVDPAURegisterVideoSurface	NO	
cuVDPAUCtxCreate	NO	
cuVDPAUGetDevice	NO	
cuEGLStreamConsumerAcquireFrame	NO	

Function	Migration Support	Diagnostic Message
cuEGLStreamConsumerConnect	NO	
cuEGLStreamConsumerConnectWithFlags	NO	
cuEGLStreamConsumerDisconnect	NO	
cuEGLStreamConsumerReleaseFrame	NO	
cuEGLStreamProducerConnect	NO	
cuEGLStreamProducerDisconnect	NO	
cuEGLStreamProducerPresentFrame	NO	
cuEGLStreamProducerReturnFrame	NO	
cuEventCreateFromEGLSync	NO	
cuGraphicsEGLRegisterImage	NO	
cuGraphicsResourceGetMappedEglFrame	NO	
cooperative_groups::coalesced_threads	YES	DPCT1119
cooperative_groups::thread_rank	YES	
cooperative_groups::this_thread	NO	
cooperative_groups::this_grid	YES	
cooperative_groups::this_multi_grid	NO	
cooperative_groups::this_thread_block	YES	
cooperative_groups::experimental::this_thread_block	YES	
cooperative_groups::tiled_partition	YES	
cooperative_groups::experimental::tiled_partition	YES	
cooperative_groups::labeled_partition	NO	
cooperative_groups::binary_partition	NO	
cooperative_groups::sync	YES	
cooperative_groups::memcpy_async	NO	
cooperative_groups::wait	NO	
cooperative_groups::wait_prior	NO	
cooperative_groups::reduce	YES	
cooperative_groups::reduce_update_async	NO	

Function	Migration Support	Diagnostic Message
cooperative_groups::experimental::reduce_update_async	NO	
cooperative_groups::reduce_store_async	NO	
cooperative_groups::experimental::reduce_store_async	NO	
cooperative_groups::plus	YES	
cooperative_groups::less	YES	
cooperative_groups::greater	YES	
cooperative_groups::bit_and	YES	
cooperative_groups::bit_or	YES	
cooperative_groups::bit_xor	YES	
cooperative_groups::inclusive_scan	YES	Partial
cooperative_groups::exclusive_scan	YES	Partial
cooperative_groups::experimental::inclusive_scan_update	NO	
cooperative_groups::inclusive_scan_update	NO	
cooperative_groups::experimental::exclusive_scan_update	NO	
cooperative_groups::exclusive_scan_update	NO	
cooperative_groups::invoke_one	NO	
cooperative_groups::invoke_one_broadcast	NO	
cooperative_groups::thread_group::size	YES	
cooperative_groups::thread_group::num_threads	YES	
cooperative_groups::thread_group::thread_rank	YES	
cooperative_groups::thread_group::sync	YES	
cooperative_groups::thread_group::get_type	YES	
cooperative_groups::thread_block::sync	YES	

Function	Migration Support	Diagnostic Message
cooperative_groups::thread_block::thread_rank	YES	
cooperative_groups::thread_block::group_index	YES	
cooperative_groups::thread_block::thread_index	YES	
cooperative_groups::thread_block::dim_threads	NO	
cooperative_groups::thread_block::num_threads	YES	
cooperative_groups::thread_block::size	YES	
cooperative_groups::thread_block::group_dim	NO	
cooperative_groups::cluster_group::sync	NO	
cooperative_groups::cluster_group::barrier_arrive	NO	
cooperative_groups::cluster_group::barrier_wait	NO	
cooperative_groups::cluster_group::thread_rank	NO	
cooperative_groups::cluster_group::block_rank	NO	
cooperative_groups::cluster_group::num_threads	NO	
cooperative_groups::cluster_group::num_blocks	NO	
cooperative_groups::cluster_group::dim_threads	NO	
cooperative_groups::cluster_group::dim_blocks	NO	
cooperative_groups::cluster_group::block_index	NO	
cooperative_groups::cluster_group::query_shared_rank	NO	
cooperative_groups::cluster_group::map_shared_rank	NO	
cooperative_groups::cluster_group::size	NO	

Function	Migration Support	Diagnostic Message
cooperative_groups::grid_group::is_valid	NO	
cooperative_groups::grid_group::sync	YES	
cooperative_groups::grid_group::thread_rank	YES	
cooperative_groups::grid_group::block_rank	YES	
cooperative_groups::grid_group::cluster_rank	NO	
cooperative_groups::grid_group::num_threads	YES	
cooperative_groups::grid_group::num_blocks	YES	
cooperative_groups::grid_group::num_clusters	NO	
cooperative_groups::grid_group::dim_blocks	NO	
cooperative_groups::grid_group::dim_clusters	NO	
cooperative_groups::grid_group::block_index	NO	
cooperative_groups::grid_group::cluster_index	NO	
cooperative_groups::grid_group::size	YES	
cooperative_groups::grid_group::group_dim	NO	
cooperative_groups::multi_grid_group::is_valid	NO	
cooperative_groups::multi_grid_group::sync	NO	
cooperative_groups::multi_grid_group::num_threads	NO	
cooperative_groups::multi_grid_group::thread_rank	NO	
cooperative_groups::multi_grid_group::grid_rank	NO	
cooperative_groups::multi_grid_group::num_grids	NO	

Function	Migration Support	Diagnostic Message
cooperative_groups::multi_grid_group::size	NO	
cooperative_groups::thread_block_tile::sync	YES	
cooperative_groups::thread_block_tile::num_threads	YES	
cooperative_groups::thread_block_tile::thread_rank	YES	
cooperative_groups::thread_block_tile::meta_group_size	NO	
cooperative_groups::thread_block_tile::meta_group_rank	YES	
cooperative_groups::thread_block_tile::shfl	YES	DPCT1119
cooperative_groups::thread_block_tile::shfl_up	YES	
cooperative_groups::thread_block_tile::shfl_down	YES	
cooperative_groups::thread_block_tile::shfl_xor	YES	
cooperative_groups::thread_block_tile::any	NO	
cooperative_groups::thread_block_tile::all	NO	
cooperative_groups::thread_block_tile::ballot	NO	
cooperative_groups::thread_block_tile::match_any	NO	
cooperative_groups::thread_block_tile::match_all	NO	
cooperative_groups::thread_block_tile::size	YES	
cooperative_groups::coalesced_group::sync	YES	DPCT1119
cooperative_groups::coalesced_group::num_threads	NO	
cooperative_groups::coalesced_group::thread_rank	YES	DPCT1119

Function	Migration Support	Diagnostic Message
cooperative_groups::coalesced_group::meta_group_size	NO	
cooperative_groups::coalesced_group::meta_group_rank	NO	
cooperative_groups::coalesced_group::shfl	YES	DPCT1119
cooperative_groups::coalesced_group::shfl_up	NO	
cooperative_groups::coalesced_group::shfl_down	NO	
cooperative_groups::coalesced_group::any	NO	
cooperative_groups::coalesced_group::all	NO	
cooperative_groups::coalesced_group::ballot	NO	
cooperative_groups::coalesced_group::match_any	NO	
cooperative_groups::coalesced_group::match_all	NO	
cooperative_groups::coalesced_group::size	YES	DPCT1119
nvrtcGetErrorString	NO	
nvrtcVersion	NO	
nvrtcCreateProgram	NO	
nvrtcDestroyProgram	NO	
nvrtcCompileProgram	NO	
nvrtcGetPTXSize	NO	
nvrtcGetPTX	NO	
nvrtcGetProgramLogSize	NO	
nvrtcGetProgramLog	NO	
nvrtcAddNameExpression	NO	
nvrtcGetLoweredName	NO	
nvrtcGetNumSupportedArchs	NO	
nvrtcGetSupportedArchs	NO	

Function	Migration Support	Diagnostic Message
nvrtcGetCUBIN	NO	
nvrtcGetCUBINSize	NO	
nvrtcGetNVVM	NO	
nvrtcGetNVVMSize	NO	
nvrtcGetTypeName	NO	
nvrtcGetLTOIR	NO	
nvrtcGetLTOIRSize	NO	
nvrtcGetOptiXIR	NO	
nvrtcGetOptiXIRSize	NO	
mul24	YES	
saturate	YES	
cuCabs	YES	
cuCabsf	YES	
cuCadd	YES	
cuCaddf	YES	
cuCdiv	YES	
cuCdivf	YES	
cuCfma	YES	
cuCfmaf	YES	
cuCimag	YES	
cuCimagf	YES	
cuCmul	YES	
cuCmulf	YES	
cuComplexDoubleToFloat	YES	
cuComplexFloatToDouble	YES	
cuConj	YES	
cuConjf	YES	
cuCreal	YES	
cuCrealf	YES	
cuCsub	YES	

Function	Migration Support	Diagnostic Message
cuCsubf	YES	
make_cuComplex	YES	
make_cuDoubleComplex	YES	
make_cuFloatComplex	YES	
__assert_fail	YES	
__assertfail	YES	
cuGetExportTable	YES	Partial

thrust API

Function	Migration Support	Diagnostic Message
thrust::abs	YES	
thrust::acos	YES	
thrust::acosh	YES	
thrust::adjacent_difference	YES	
thrust::advance	YES	
thrust::make_transform_output_iterator	YES	
thrust::all_of	YES	
thrust::any_of	YES	
thrust::arg	NO	
thrust::asin	YES	
thrust::asinh	YES	
thrust::atan	YES	
thrust::atanh	YES	
thrust::binary_search	YES	
thrust::conj	YES	
thrust::copy	YES	
thrust::copy_if	YES	
thrust::copy_n	YES	
thrust::cos	YES	
thrust::cosh	YES	

Function	Migration Support	Diagnostic Message
thrust::count	YES	
thrust::count_if	YES	
thrust::device_delete	YES	
thrust::device_free	YES	
thrust::device_malloc	YES	
thrust::device_new	YES	
thrust::device_pointer_cast	YES	
thrust::distance	YES	
thrust::equal	YES	
thrust::equal_range	YES	
thrust::exclusive_scan	YES	
thrust::exclusive_scan_by_key	YES	
thrust::exp	YES	
thrust::fill	YES	
thrust::fill_n	YES	
thrust::find	YES	
thrust::find_if	YES	
thrust::find_if_not	YES	
thrust::for_each	YES	
thrust::for_each_n	YES	
thrust::free	YES	
thrust::gather	YES	
thrust::gather_if	YES	
thrust::generate	YES	
thrust::generate_n	YES	
thrust::get	YES	
thrust::get_temporary_buffer	YES	
thrust::inclusive_scan	YES	
thrust::inclusive_scan_by_key	YES	
thrust::inner_product	YES	

Function	Migration Support	Diagnostic Message
thrust::is_partitioned	YES	
thrust::shuffle	NO	
thrust::shuffle_copy	NO	
thrust::is_sorted	YES	
thrust::is_sorted_until	YES	
thrust::log	YES	
thrust::log10	YES	
thrust::lower_bound	YES	
thrust::make_counting_iterator	YES	
thrust::make_discard_iterator	YES	
thrust::make_pair	YES	
thrust::make_permutation_iterator	YES	
thrust::make_reverse_iterator	YES	
thrust::make_transform_iterator	YES	
thrust::make_tuple	YES	
thrust::tie	YES	
thrust::make_zip_iterator	YES	
thrust::malloc	YES	
thrust::max_element	YES	
thrust::max	YES	
thrust::min	YES	
thrust::merge	YES	
thrust::merge_by_key	YES	
thrust::min_element	YES	
thrust::minmax_element	YES	
thrust::mismatch	YES	
thrust::none_of	YES	
thrust::norm	YES	
thrust::not1	YES	
thrust::not2	YES	

Function	Migration Support	Diagnostic Message
thrust::operator!=	YES	
thrust::operator*	YES	
thrust::operator+	YES	
thrust::operator-	YES	
thrust::operator/	YES	
thrust::operator<	YES	
thrust::operator<<	YES	
thrust::operator<=	YES	
thrust::operator==	YES	
thrust::operator>	YES	
thrust::operator>=	YES	
thrust::operator>>	NO	
thrust::partition	YES	
thrust::partition_copy	YES	
thrust::partition_point	YES	
thrust::polar	YES	
thrust::pow	YES	
thrust::proj	YES	
thrust::raw_pointer_cast	YES	
thrust::raw_reference_cast	YES	
thrust::reduce	YES	
thrust::reduce_by_key	YES	
thrust::reinterpret_tag	NO	
thrust::remove	YES	
thrust::remove_copy	YES	
thrust::remove_copy_if	YES	
thrust::remove_if	YES	
thrust::replace	YES	
thrust::replace_copy	YES	
thrust::replace_copy_if	YES	

Function	Migration Support	Diagnostic Message
thrust::replace_if	YES	
thrust::retag	NO	
thrust::return_temporary_buffer	YES	
thrust::reverse	YES	
thrust::reverse_copy	YES	
thrust::scatter	YES	
thrust::scatter_if	YES	
thrust::sequence	YES	
thrust::set_difference	YES	
thrust::set_difference_by_key	YES	
thrust::set_intersection	YES	
thrust::set_intersection_by_key	YES	
thrust::set_symmetric_difference	YES	
thrust::set_symmetric_difference_by_key	YES	
thrust::set_union	YES	
thrust::set_union_by_key	YES	
thrust::sin	YES	
thrust::sinh	YES	
thrust::sort	YES	
thrust::sort_by_key	YES	
thrust::sqrt	YES	
thrust::stable_partition	YES	
thrust::stable_partition_copy	YES	
thrust::stable_sort	YES	
thrust::stable_sort_by_key	YES	
thrust::swap	YES	
thrust::swap_ranges	YES	
thrust::system_category	NO	
thrust::tabulate	YES	
thrust::tan	YES	

Function	Migration Support	Diagnostic Message
thrust::tanh	YES	
thrust::THRUST_PREVENT_MACRO_SUBSTITUTION	NO	
thrust::transform	YES	
thrust::transform_exclusive_scan	YES	
thrust::transform_if	YES	
thrust::transform_inclusive_scan	YES	
thrust::transform_reduce	YES	
thrust::uninitialized_copy	YES	
thrust::uninitialized_copy_n	YES	
thrust::uninitialized_fill	YES	
thrust::uninitialized_fill_n	YES	
thrust::unique	YES	
thrust::unique_by_key	YES	
thrust::unique_by_key_copy	YES	
thrust::unique_copy	YES	
thrust::upper_bound	YES	
thrust::unique_count	YES	
thrust::detail::vector_equal	YES	
thrust::system::system_category	YES	
thrust::uniform_int_distribution::min	YES	
thrust::uniform_int_distribution::max	YES	
thrust::host_vector::size	YES	
thrust::device_vector::size	YES	
thrust::device_ptr::get	YES	

WMMA API

Function	Migration Support	Diagnostic Message
nvcuda::wmma::fill_fragment	YES	
nvcuda::wmma::load_matrix_sync	YES	
nvcuda::wmma::store_matrix_sync	YES	
nvcuda::wmma::mma_sync	YES	
nvcuda::wmma::bmma_sync	NO	

Frequently Asked Questions

General Information

- How do I migrate source files that use C++20 or newer standard features on Linux* and Windows*?
- How do I migrate files on Windows when using a CMake project?
- How is the migrated code formatted?
- Why does the compilation database not contain all source files in the project?
- How do I use the migrated module file in the new project?
- Is the memory space allocated by sycl::malloc_device, sycl::malloc_host, and dpct::dpct_malloc initialized?
- How do I migrate CUDA* source code that contains CUB library implementation source code?
- How do I fix the issue of SYCL* code hanging due to work group level synchronization, such as a group barrier used in a conditional statement?

Troubleshooting Migration

- How do I fix an error such as "error: unknown type name" when I migrate files with "dpct –in-root=srcdir –out-root=dstdir *.cu"?
- How do I fix a parsing error such as "no member named 'max' in namespace 'std'" or "no member named 'min' in namespace 'std'" when migrating code on Windows?
- How do I fix a compilation error such as "error: dlopen not declared" when I compile code on a Windows machine, that was originally migrated on Linux?
- Why didn't the "atomic*" APIs get migrated?
- Why did my migration fail with "error: restrict requires a pointer or reference"?
- How do I resolve incorrect runtime behavior for dpct::dev_mgr and dpct:mem_mgr in a library project that is loaded more than once in another application?
- Why do I get "warning: shift count >= width of type" when I compile migrated code with the Intel® oneAPI DPC++/C++ Compiler?
- How do I resolve missing include errors that occur when migrating my code?

General Information

How do I migrate source files that use C++20 or newer standard features on Linux* and Windows*?

On Linux, the default C++ standard for Intel® DPC++ Compatibility Tool's parser is C++17. If you want to enable newer standard features in Intel® DPC++ Compatibility Tool, you need to add the `--extra-arg="-std=<value>"` option to the command line. The supported values are:

- `c++14`
- `c++17`
- `c++20`
- `c++23`

- `c++26`

On Windows, the default C++ standard for Intel® DPC++ Compatibility Tool's parser is C++17. If you want to enable C20 features in Intel® DPC++ Compatibility Tool, you need to add the option `--extra-arg="-std=c++20"` to the command line.

How do I migrate files on Windows when using a CMake project?

For a CMake project on a Windows OS, you can use CMake to generate Microsoft Visual Studio* project files (`vcxproj` files). Then choose one of the following options:

- Migrate the source files on the command line by using the `--vcxprojfile` option of Intel® DPC++ Compatibility Tool.
- Migrate the entire project in Microsoft Visual Studio with an Intel® DPC++ Compatibility Tool Microsoft Visual Studio plugin.

How is the migrated code formatted?

Intel® DPC++ Compatibility Tool provides two options to control the format of migrated code: `--format-range` and `--format-style` .

If input source code is well formatted, Intel® DPC++ Compatibility Tool will use default options settings `--format-range` and `--format-style` to format the resulting code.

If input source code is not well formatted (for example, the tool detects mixed use of tabs and spaces or mixed indents) you can do one of the following:

- Intel® DPC++ Compatibility Tool will try to detect the indent size of the original code and apply it to the resulting code. You can guide the tool by setting `TabWidth` and `UseTab` in the `.clang-format` file. Because the input source code is not well formatted, the indents in the resulting code may still be inconsistent.
- Run Intel® DPC++ Compatibility Tool with the `--format-range=all` option to format the entire resulting file. The change between input source code and resulting source code may be large and make it more difficult to compare the code.
- Format your input source code, then use Intel® DPC++ Compatibility Tool with the same `.clang-format` file for migration.

Why does the compilation database not contain all source files in the project?

In the project build folder, the command `intercept-build make [target]` is used to generate the compilation database. The content of the compilation database depends on the optional [target] parameter. If you need to get the list of files corresponding to default build target, do not specify the [target] parameter.

Make sure to disable ccache (compiler cache) in your project before using intercept-build. If ccache is enabled, intercept-build cannot generate the complete compilation database as some compile commands may be skipped if the target objects are already available in the cache. Use the following command to disable ccache before running the intercept-build command:

```
export CCACHE_DISABLE=1
```

How do I use the migrated module file in the new project?

`.cu` module files are compiled with the `-ptx` or `-cubin` options in the original project and dynamically loaded into other `*.cu` files with `cuModuleLoad()` or `cuModuleLoadData()`.

Intel® DPC++ Compatibility Tool migrates module file code in the same way as other `*.cu` files. In addition, it adds a wrapper function for each function in the module file that has the `_global_` attribute.

You can compile the migrated module file into a dynamic library and load the library with a dynamic library API appropriate to your platform. For example:

- In Linux, load a dynamic library (`.so`) using `dlopen()`
- In Windows, load a dynamic library (`.dll`) using `LoadLibraryA()`

Is the memory space allocated by sycl::malloc_device, sycl::malloc_host, and dpct::dpct_malloc initialized?

The memory allocated by `sycl::malloc_device`, `sycl::malloc_host`, and `dpct::dpct_malloc` is not initialized. If your program explicitly or implicitly relies on the initial value of newly allocated memory, the program may fail at runtime. Adjust your code to avoid such failures.

For example, the following original code:

```
// original code

int *device_mem = nullptr;device_mem = sycl::malloc_device<int>(size, dpct::get_default_queue());
device_mem[0] += somevalue;
```

is adjusted to initialize the newly allocated memory to 0 before use:

```
// fixed SYCL code

int *device_mem = nullptr;device_mem = sycl::malloc_device<int>(size, dpct::get_default_queue());
dpct::get_default_queue().memset(0, size).wait();
device_mem[0] += somevalue;
```

How do I migrate CUDA* source code that contains CUB library implementation source code?

If you migrate the CUB library implementation code directly, you may not get the expected results. Instead, exclude CUB library implementation source code from your migration by adding `--in-root-exclude=<path to CUB library source code>` to your migration command.

How do I fix the issue of SYCL* code hanging due to work group level synchronization, such as a group barrier used in a conditional statement?

If synchronization API in control flow statements like a conditional statement and loop statement are called in SYCL code, you may encounter a runtime hang issue. The basic idea to fix the hang issue is to ensure that each synchronization API is either reached by all work items of a workgroup, or skipped by all the work items of a workgroup.

Here are two examples of how to fix:

In the first example, the synchronization API group barrier(nd_item.barrier()) is called inside an if block. The evaluation results of the conditional statement differ in each work item so not all work items can reach the synchronization API.

```
// original code

void kernel(const sycl::nd_item<3> &item_ct1) {
   unsigned int tid = item_ct1.get_local_id(2);
   if (tid < 32) {
      // CODE block 1
      ...
      item_ct1.barrier(sycl::access::fence_space::local_space);
      // CODE block 2
      ...
   }
}
```

The following code shows how to fix the hang issue by moving the synchronization statement out of the if block.

```
// fixed SYCL code

void kernel(const sycl::nd_item<3> &item_ct1) {
   unsigned int tid = item_ct1.get_local_id(2);
   if (tid < 32) {
      // CODE block 1
      ...
   }
   item_ct1.barrier(sycl::access::fence_space::local_space);
   if (tid < 32) {
      // CODE block 2
      ...
   }
}
```

The second example demonstrates how to fix the hang issue when a synchronization API is used in a for loop:

```
// original code

void compute(int id_space, const sycl::nd_item<3> &item_ct1) {
   unsigned int id = item_ct1.get_group(2) * item_ct1.get_local_range(2) +
item_ct1.get_local_id(2);
   for (; id < id_space; id += item_ct1.get_group_range(2) * item_ct1.get_local_range(2)) {
      ...
      item_ct1.barrier();
      ...
   }
}
```

The following code shows how to fix the hang issue by making sure all work items have same run footprint in the for loop.

```
// fixed SYCL code

void compute(int id_space, const sycl::nd_item<3> &item_ct1) {
   unsigned int id = item_ct1.get_group(2) * item_ct1.get_local_range(2) +
item_ct1.get_local_id(2);
   unsigned int num_workitem = item_ct1.get_group_range(2) * item_ct1.get_local_range(2);
   // The condition is updated to make sure all work items can enter the loop body in each
iteration
   for (; id < ((id_space + num_workitem - 1) / num_workitem) * num_workitem;
   id += item_ct1.get_group_range(2) * item_ct1.get_local_range(2)) {
      ...
      item_ct1.barrier();
      ...
   }
}
```

Troubleshooting Migration

How do I fix an error such as “error: unknown type name” when I migrate files with “dpct –in-root=srcdir –out-root=dstdir *.cu”?

The problem may be caused by files in the `*.cu` list, which can be used as header files (included with an `#include` statement) and are not supposed to be parsed as a standalone file. In this case, Intel® DPC++ Compatibility Tool reports an error if it cannot parse the file because the file depends on the definitions/ declarations in other files. Use one of the methods below to migrate your content:

- Rely on Intel® DPC++ Compatibility Tool to decide which files to migrate with: `compile_commands.json: "dpct -p=compile_commands.json --in-root=srcdir --out-root=dstdir"`
- Manually pass specific files to migrate, but do not pass the files that are included in other files and not supposed to be compiled as a standalone file in the original application. The header files are migrated automatically when they are included by the files provided as the input to the tool and are located within the `in-root` folder: `dpct --in-root= srcdir --out-root=dstdir sample.cu`

How do I fix a parsing error such as "no member named 'max' in namespace 'std'" or "no member named 'min' in namespace 'std'" when migrating code on Windows?

Use one of the following methods to resolve the error:

- Add `#include <algorithm>` to the source file before using `std::min` and `std::max`
- Define the NOMINMAX macro by inserting `#define NOMINMAX` before including `WinDef.h`

How do I fix a compilation error such as "error: dlopen not declared" when I compile code on a Windows machine, that was originally migrated on Linux?

When Intel® DPC++ Compatibility Tool generates the source code, it uses dynamic loading APIs specific to the OS on which Intel® DPC++ Compatibility Tool is running.

For example, `dlopen`, `dlclose`, and `dlsym` are used on Linux and `LoadLibraryA`, `FreeLibrary`, and `GetProcAddress` are used on Windows.

If your code was migrated on a OS that is different from the OS you need to compile the generated code on, migrate the project again with the Intel® DPC++ Compatibility Tool on the target OS or fix the code manually.

Why didn't the "atomic*" APIs get migrated?

Intel® DPC++ Compatibility Tool may assume that the "atomic*" APIs are user-defined APIs, in which case they are not migrated.

This can occur in the following scenarios:

- The CUDA include path is specified by both `--cuda-include-path` and `-I*`, but the paths are different
- The CUDA include path is specified by `-I*`, but there are other CUDA include files located on the default CUDA install path

To make sure "atomic*" APIs are migrated, don't use `-I*` to specify the CUDA include path with the `dpct` migration command. Instead, use only `--cuda-include-path` to specify the CUDA include path.

Why did my migration fail with "error: restrict requires a pointer or reference"?

The C++ standard does not support the restrict qualifier and the C standard supports the restrict qualifier only on pointers to an object type.

Based on these language standards Intel® DPC++ Compatibility Tool emits the parsing error.

You may need to adjust the source code.

How do I resolve migration failure with "fatal error: 'cmath' file not found" in Linux?

The problem stems from an absent include path for the new standard C++ library. The Intel® DPC++ Compatibility Tool is designed to automatically detect the appropriate version of the C++ header file by checking the compiler package at `/usr/lib/gcc/x86_64-linux-gnu``and C++ header at ``/usr/include/c++`. In the following example, it tries to use C++ header version 12 based on the knowledge of the compiler package, but it fails because C++ header version 12 does not exist."

```
ls /usr/lib/gcc/x86_64-linux-gnu
11 12
ls /usr/include/c++
11
```

To fix this issue, please install the version 12 g++ package or libstdc++ package.

```
sudo apt install g++-12
or
sudo apt install libstdc++-12-dev
```

If your installation differs, install the missing version of `g++-XX` or `libstdc++-XX-dev` based on what you see missing from the results of "ls /usr/lib/gcc/x86_64-linux-gnu" and "ls /usr/include/c++".

How do I resolve incorrect runtime behavior for dpct::dev_mgr and dpct:mem_mgr in a library project that is loaded more than once in another application?

`dpct::dev_mgr` and `dpct::mem_mgr` are singleton classes in the Intel® DPC++ Compatibility Tool helper functions. When the helper function headers are used to build an executable project, both `dpct::dev_mgr` and `dpct::mem_mgr` will have only one instance in the executable. However, when the helper function headers are used to build a library project and the library project is loaded more than once with `dlopen()` (or `LoadLibraryA()` for Windows) in an application, more than two instances of `dpct::dev_mgr` and `dpct::mem_mgr` will be created and result in incorrect runtime behavior.

For example, both files `libA.cpp` and `libB.cpp` include Intel® DPC++ Compatibility Tool helper function header `dpct.hpp`, and they are built into dynamic libraries `libA.so` and `libB.so` respectively. If an application `main.cpp` imports the libraries with `dlopen()`, there will be two instances of `dpct::dev_mgr` and `dpct::mem_mgr` in the runtime of the application.

To resolve this issue, separate the implementation and the declaration of `dpct::dev_mgr` and `dpct::mem_mgr` in Intel® DPC++ Compatibility Tool helper function:

1. Create a new C++ file `dpct_helper.cpp`.
2. Move the implementation of `instance()` in `class dev_mgr` from `dpct/device.hpp` to `dpct_helper.cpp`.

   For example, the original `dpct/device.hpp`:

```
class dev_mgr {
public:
  static dev_mgr &instance() { // the implementation and the declaration of dev_mgr::instance
    static dev_mgr d_m;
    return d_m;
  }
  ...
}
```

   is updated to:

```
class dev_mgr {
public:
  static dev_mgr &instance();//the declaration of dev_mgr::instance
  ...
}
```

and the new `dpct_helper.cpp` now contains the implementation of `dev_mgr::instance()`:

```
#include <dpct/device.hpp>
dpct::dev_mgr &dev_mgr::instance(){ // the implementation of dev_mgr::instance
  static dev_mgr d_m;
  return d_m;
}
```

3. Similar to step two, move the implementation of `instance()` in the `class mem_mgr` from `dpct/memory.hpp` to `dpct_helper.cpp`.
4. Build `dpct_helper.cpp` into a dynamic library `libdpct_helper`.
   - In Linux:

```
dpcpp -g -shared -o libdpct_helper.so -fPIC ./dpct_helper.cpp
```

   - In Windows:

```
cl.exe /LD dpct_helper.cpp
```

5. Add library `libdpct_helper` to the environment variables.
   - In Linux: Add the location of `libdpct_helper.so` into `LD_LIBRARY_PATH`.
   - In Windows: Add the location of `libdpct_helper.dll` into `PATH`.
6. Dynamically link `libdpct_helper` when building libraries and applications.

After performing the update steps, all the libraries and applications will share the same instance of the device manager `dpct::dev_mgr` and the memory manager `dpct::mem_mgr` in Intel® DPC++ Compatibility Tool helper functions.

Why do I get “warning: shift count >= width of type” when I compile migrated code with the Intel® oneAPI DPC++/C++ Compiler?

Shifting bits where the shift is greater than the type length is undefined behavior for the Intel® oneAPI DPC++/C++ Compiler and may result in different behavior on different devices. Adjust your code to avoid this type of shift.

For example, the migrated SYCL* code:

```
// migrated SYCL code

void foo() {
  ...
  unsigned int bit = index[tid] % 32;
  unsigned int val = in[tid] << 32 - bit;
  ...
}
```

is adjusted to avoid a bit shift that is greater than the type length:

```
// fixed SYCL code

void foo() {
  ...
  unsigned int bit = index[tid] % 32;
  unsigned int val;
  if(32 - bit == 32)
    val = 0;
  else
    val = in[tid] << 32 - bit;
  ...
}
```

How do I resolve missing include errors that occur when migrating my code?

Use the option `--extra-arg=-v` to prompt Intel® DPC++ Compatibility Tool to use verbose output, which includes information about which paths the tool searches for includes.

You can provide an additional path to look for includes in one of the following ways:

- Use the `--extra-arg="-I<extra include path>"` option in your migration command to specify an additional path for the tool to use when searching for includes during migration.
- If you are using a compilation database, add the `-I<extra include path>` option to the compile command in the database for the source files, to specify the include path.

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others. SYCL is a registered trademark of the Kronos Group, Inc.

Optimization Notice
Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex. Notice revision #20201201

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.