Optimization Guide

Intel® Xeon® Scalable Processor

Data Compression with Intel® ISA-L/ Intel® IPP/ Intel® QAT Tuning Guide

Contents

Revision Record

Date	Rev.	Description
05/28/2021	1.0	Initial public release.

1. Introduction

This guide is targeted towards users who are already familiar with data compression and provides pointers and system setting for hardware and software that will provide the best performance for most situations. However, please note that we rely on the users to carefully consider these settings for their specific scenarios, since data compression can be deployed in multiple ways and this is a reference to one such use-case.

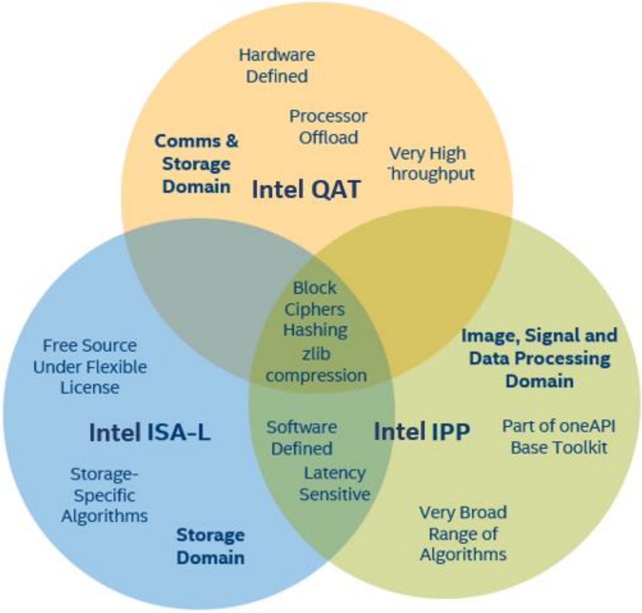
As 5G, IoT, AI, and other technologies continue to develop, the generation and usage of data has experienced exponential growth, and data transmission, storage and processing must be faster than ever. Data compression can reduce the size of data significantly, and it plays a vital role in data transmission and storage. Intel® Intelligent Storage Acceleration Library (Intel® ISA-L), Intel® Integrated Performance Primitives (Intel® IPP) and Intel® QuickAssist Technology (Intel® QAT) provide data compression functionality and can effectively improve compression performance.

Intel Intelligent Storage Acceleration Library helps accelerate and optimize Intel architecture-based storage, and provides optimizations for storage recoverability, data integrity, data security, among others, while accelerating data compression. For more information on ISA-L, please visit Intel Storage Acceleration Library.

Intel Integrated Performance Primitives is a cross-platform software function library that provides a wide range of image, signal, and data processing functions. Intel IPP is optimized for the characteristics of CPU, helping users to better utilize the latest features (such as AVX-512) on the IA platform to improve application performance. For more information on Intel IPP please visit Intel Integrated Performance Primitives.

Intel Quick Assist Technology is a high-performance data security and compression acceleration solution provided by Intel. This solution utilizes the QAT chip to share symmetrical/asymmetrical encryption computations, DEFLATE lossless compression, and other computation intensive tasks for lower CPU utilization and higher overall platform performance. For more information on Intel QAT please visit Intel QuickAssist Technology.

Intel ISA-L/ Intel IPP/ Intel QAT focus on different areas and provide different functions. Intel ISA-L focuses on storage and provides a series of erasure coding/data protection, data integrity/compression/hashing/encryption and other low-level optimization functions. Intel IPP focuses on signal, image, and data processing, and utilizes SIMD (Single Instruction, Multiple Data) instructions to increase the performance of computation intensive applications, including image processing, data compression, signal processing, and cryptography. Intel QAT improves the security and compression performance of cloud, network, big data, storage, and other applications, by offloading computation intensive operations from CPU to Intel QAT devices. Intel ISA-L/IPP/QAT have some overlapping and unique functions, as shown in the figure below.


Hardware Defined
Processor Offload
Comms & Storage Domain
Intel QAT
Very High Throughput
Block Ciphers Hashing zlib compression
Free Source Under Flexible License
Image, Signal and Data Processing Domain
Intel ISA-L
Software Defined
Intel IPP
Part of oneAPI Base Toolkit
Latency Sensitive
Storage-Specific Algorithms
Storage Domain
Very Broad Range of Algorithms

3rd Gen Intel® Xeon® Scalable processors deliver industry-leading, workload-optimized platforms with built-in AI acceleration, providing a seamless performance foundation to help speed data's transformative impact, from the multi-cloud to the intelligent edge and back. Improvements of particular interest to this workload applications are:

- Enhanced Performance
- More Intel® Ultra Path Interconnect
- Increased DDR4 Memory Speed & Capacity
- Intel® Advanced Vector Extensions
- Intel® Security Essentials and Intel® Security Libraries for Data Center
- Intel® Speed Select Technology

Tested hardware and software environment for this tuning guide:

Server Configuration	**Hardware**	CPU	Intel® Xeon® PLATINUM 8360Y CPU @ 2.20GHz
		Memory	16*32 GB DDR4, 3200 MT/s
		Storage/Disks	Intel SSD S4610, 960G
		NIC	Intel® Ethernet Controller XXV700 25GbE SFP28
		Intel QAT	Intel® QuickAssist Adapter 8970
	Software	Operating System	CentOS* 7.8
		Kernel	3.10.0-1127.el7.x86_64
		Zlib*	zlib-1.2.7.3
		Intel ISA-L	isa-l-v2.30.0
		Intel IPP	oneAPI 2021.1.1
		QAT Driver	qat1.7.1.4.12.0
		QATzip	QATzip-1.0.2

Note: The configuration described in this article is based on 3rd Generation Intel Xeon processor hardware. Server platform, memory, hard drives, network interface cards can be determined according to customer usage requirements.

2. BIOS Settings

The configuration items that can be optimized for BIOS and the recommended non-default values are as follows:

Configuration item	Recommended value
Advanced/Power & Performance/CPU Power and Performance Policy	Performance
Advanced/Power & Performance/CPU C State Control	Package C State = C0/C1

	C1E = Disabled Processor C = Disabled
Advanced/Power & Performance/ CPU P State Control/Enhanced Intel SpeedStep® Tech	Disabled

2.1. CPU Power and Performance Policy

This setting allows the user to set an overall power and performance policy for the system, and when changed will modify a selected list of options to achieve the policy. These options are still changeable outside of the policy but do reflect the changes that the policy makes when a new policy is selected. The Performance setting is strongly weighted towards performance, even at the expense of energy efficiency.

2.2. CPU State Control

The CPU C State Control screen allows the user to specify a policy, which is optimized for the processor sleep state. Sets and specifies the lowest C-state for Processor package.

- C0/C1 state is no package C-state support.
- C6 retention state provides more power savings.
- C1E when enabled the CPU will switch to the Minimum Enhanced Intel SpeedStep® Technology operating point when all execution cores enter C1. Frequency will switch immediately, followed by gradual voltage switching.
- Processor C6 setting enables/disables processor C6 (ACPI C3) report to OS.

2.3. Enhanced Intel SpeedStep® Tech

Enhanced Intel SpeedStep Technology allows the system to dynamically adjust processor voltage and core frequency, which can result in decreased average power consumption and decreased average heat production. When disabled, the processor setting reverts to running at maximum thermal design power (TDP) core frequency (rated frequency).

3. Linux* Optimizations

3.1. Network Configuration

Performance in data compression applications is usually restricted by network bandwidth. Therefore, cross-network operations perform better with higher NIC bandwidth. We recommend a bandwidth of 10GB/s and higher.

3.2. CPU Configuration

- Set the CPU to the performance mode

```
cpupower -c <cpulist> frequency-set --governor performance
```

- Set the energy/performance bias

```
x86_energy_perf_policy performance
```

4. Installing and Using Intel Intelligent Storage Acceleration Library

4.1. Installing Intel ISA-L

Some of the low-level core functions of Intel ISA-L are optimized using SIMD. Intel ISA-L also provides optimization functions for CPUs that do not support or support only a small number of SIMD instructions. It has been tuned in many details with CRC, DEFLATE, and Huffman coding optimizations for compression algorithms. It also supports multi-threading and offers great performance improvements over zlib*. The compression output of Intel ISA-L is completely compatible with the gzip* format, even if there is no change upon decompression.

The standard Linux installation from source code method is used for the compilation and installation of Intel ISA-L.

1) Download source code from Github.

2) Compilation commands shown below. For detailed information, refer to the README file.

```
./autogen.sh
./configure
make
sudo make install
```

4.2. Using Intel ISA-L

Intel ISA-L provides the igzip executable program, which is used in a similar way to gzip. Intel ISA-L supports the RFC 1951 data compression algorithm and provides its own compression/decompression interfaces. It supports 4 levels (level [0,3]) of compression. The higher the level, the greater the compression ratio, with 3 being the highest.

```
$ igzip -h
Usage: igzip [options] [infiles]

Options:
 -h, --help        help, print this message
 -#            use compression level # with 0 <= # <= 3
 -o  <file>       output file
 -c, --stdout       write to stdout
 -d, --decompress   decompress file
 -z, --compress     compress file (default)
...
```

Intel ISA-L provides a performance test program. The “***make other***” command can be used to generate an ***igzip/igzip_perf*** performance test program which can test Zlib’s and Intel ISA-L’s compression performance respectively. To use the **igzip_perf** for testing, the basic command is “**igzip_perf -l 1 file**”, where the “**-z**” parameter is used to test the zlib performance.

```
./igzip_perf -h
Usage: igzip_perf [options] <infile>
  -h         help, print this message
  The options -l, -f, -z may be used up to 32 times
  -l <level>    isa-l stateless deflate level to test (0-3)
  -f <level>    isa-l stateful deflate level to test (0-3)
  -z <level>    zlib  deflate level to test
  -d <time>    approx time in seconds for deflate (at least 0)
  -i <time>     approx time in seconds for inflate (at least 0)
  -s         performance test isa-l stateful inflate
  -t         performance test isa-l stateless inflate
  -u         performance test zlib inflate
...
```

5. Installing and Using Intel Integrated Performance Primitives

5.1. Installing Intel IPP

Intel IPP is a component of the Intel® oneAPI Base Toolkit and can be installed via oneAPI.

1) Download the installation package. Select your operating system and installation method before you download.

2) Install

```
sh ./<installer>.sh
```

Intel IPP will be installed to the path: **/opt/intel/oneapi/ipp**.

3) Integrate Intel IPP and ZLib*

Unlike Intel ISA-L, the Zlib function of IPP does not provide a complete DEFLATE compression/decompression function, but instead optimizes Zlib's core functions so that Zlib runs faster. IPP provides patches for different Zlib versions. Enabling the "**-DWITH_IPP**" option while compiling Zlib will result in a Zlib library with IPP optimizations. For more details on how to compile Intel IPP Zlib, please refer to the Readme file under the "**components/interfaces/ipp_zlib**" directory of the IPP product. The main steps are:

Download the Zlib source code and decompress it.

Use the IPP patch: "patch -p1 < zlib-1.2.7.3.patch"

```
                export CFLAGS="-m64 -DWITH_IPP -I$IPPROOT/include"
                ./configure
      make shared
```

Replace the system's original **libz.so** file with the compiled **libz.so** file

5.2. Using Intel IPP Zlib

In terms of compatibility, IPP is fully integrated into the open source Zlib and is 100% compatible with the traditional Zlib library. Intel IPP Zlib guarantees binary compatibility, and so Intel IPP Zlib can be used to replace the traditional Zlib.

Intel IPP provides the test program, **zpipe.c**, which can be found in the "**zlib-<version>/examples**" directory and the following command can be used to link it to the IPP library.

```
gcc -O3 -o zpipe_ipp.out zpipe.c -I$IPPROOT/include $HOME/zlib-1.x.xx/libz.a -L$IPPROOT/lib/intel64 -lippdc
-lipps -lippcore
```

As a comparison, the following command is used to link it to the original zlib.

```
gcc -O3 -o zpipe.out zpipe.c -lz
```

The command to compress the file is:

```
./zpipe_ipp.out < input > output
```

The command to decompress the file is:

```
./zpipe_ipp.out -d < input > output
```

Reference for Intel IPP parameter settings:

1. If the fastest compression speed is required, the compression level can be set to -2. The compression ratio will be slightly smaller than Level 1, but with very good performance.

2. Compression Levels 1 to 9 are compatible with those of Zlib, while Levels 11 to 19 are similar to Levels 1 to 9, and suitable for compression of larger files (>1MB).

5.3. Compression Performance

Intel ISA-L is very high in performance, about 5 times that of Zlib. Intel IPP also has a very significant performance improvement over Zlib. These methods can be chosen according to specific requirements.

- If a higher performance and a smaller compression ratio is required, Intel ISA-L can be used.
- If a higher compression ratio is required at the cost of some performance, Intel IPP can be used.

Attribute	Intel® ISA-L	Intel® IPP Zlib
Compression/decompression speed	Very high	high
Compress ratio	Medium	Medium to high
Ease of use	Yes, as it calls ISA-L directly	Yes, if the standard Zlib interface is used. Simply replace it if needed.
Standard Zlib API source code/binary compatibility	No	Yes
Open Source	Yes	No

6. Installing and Using Intel QuickAssist Technology

Intel® QuickAssist Accelerator technology is built into some Intel adapter PCIe cards, which need to be installed into the server's PCIe slot. Some Intel QAT accelerator cards are shown below.

Intel QAT supports hardware-accelerated Deflate lossless compression algorithms. Since data compression is a computation intensive task, compression usually greatly increases CPU overheads. Using Intel QAT enables transparent data compression/decompression without increasing CPU overheads, saving disk space and disk read and write bandwidth, while increasing disk read and write throughput (IOPS).

Intel QAT compression can be used with many applications such as Nginx*, Hadoop*, ZFS*, and RocksDB*, for considerably better application performance. For more details, please refer to the QuickAssist Technology webpage..

6.1. Installing Intel QAT

The Intel QAT driver can be downloaded from page mentioned in the previous paragraph, and installed by following the standard "**./configure; make install**" command. Sample programs can be installed by using the "**make samples-install**". The sample program "**cpa_sample_code runTests=32**" can be used to test compression performance.

After installation, the **lsmod** command can be used to check the installation status. The output should be similar to the following:

```
# lsmod | grep qa
```

```
qat_c62x           18061  0
intel_qat          214941  2     usdm_drv,qat_c62x
uio               19338  1     intel_qat
```

If **qat_service** is also installed in the system, the service status can be checked.

```
# service qat_service status
Checking status of all devices.
There is 3 QAT acceleration device(s) in the system:
 qat_dev0 - type: c6xx, inst_id: 0, node_id: 1, bsf: 0000:cc:00.0,  #accel: 5 #engines: 10 state: up
qat_dev1 - type: c6xx, inst_id: 1, node_id: 1, bsf: 0000:ce:00.0,  #accel: 5 #engines: 10 state: up
 qat_dev2 - type: c6xx, inst_id: 2, node_id: 1, bsf: 0000:d0:00.0,  #accel: 5 #engines: 10 state: up
```

6.2. Intel QAT Settings

The interface between the Intel QAT accelerator card driver and hardware is implemented by "**ring**" - "**request ring**" and "**response ring**". The driver uses a **request ring** to send data to the accelerator card, and a **response ring** is used to receive the response data. Whether the response data is usable can be identified using "**interrupt**", or by polling the **response ring**.

For best performance, the following factors need to be considered:

1) Intel QAT supports 3 programming modes: Interrupt, Polling and Epoll. If the data flow is fairly stable (such as package processing applications) the Polling mode is recommended, with the off-loading cost being the lowest. If the data flow has sudden spikes, the Epoll mode is recommended so that the application can "sleep" until there is a response that requires processing.
   When using the Polling mode:
   - The polling cycle needs to be carefully adjusted for best performance
   - It is best to call the polling API and submit new requests using the same thread

   When using the Epoll mode:
   - When the CPU is idle, the CPU utilization should be 0%. It should not be 0 in any other statuses. Polling may be better than Epoll in case of high workloads.
2) Synchronous and asynchronous: The Intel QAT API supports both synchronous and asynchronous operation modes. For the best performance, applications should be able to submit multiple pending requests to the acceleration engine, greatly reducing delays in processing requests on the engine. Pending requests can be submitted from the application using asynchronous submission requests or using the synchronous mode in multi threads. As multi-threading creates additional overheads due to context switching between threads, the asynchronous mode is recommended.
3) Avoid PCI performance bottlenecks: the following prerequisites are required for best performance:
   - Data buffer should be 64-byte aligned.
   - The data transmission size should be an integer multiple of 64 bytes.
   - Avoid transmitting data that is less than 64 bytes.
4) Disable parameter checking: Parameter checking increases the clock cycle the driver uses. It is enabled by default and can be enabled/disabled using the **ICP_PARAM_CHECK** environment variable.

For more information about Intel QAT performance optimization, refer to “Intel® QuickAssist Technology - Performance Optimization Guide”.

Intel QAT Configuration File

The Intel QAT driver uses a configuration file to configure operating parameters. The default configuration file (in the "**quickassist/utilities/adf_ctl/conf_files**" directory) is included in the software package. Using the Intel C62x chipset as an example, the configuration file is "**c6xx_dev[x].conf**", which can be edited as required. The following is a sample configuration file and its descriptions:

```
[GENERAL]
ServicesEnabled = cy;dc            # enabled services, cy means encryption, dc means compression
...
[KERNEL]
                         ...
[SHIM]
NumberCyInstances = 0   # number of encrypted instances
NumberDcInstances = 1   # number of compression instances
NumProcesses = 32      # number of user space processes
# Data Compression - User instance #0
Dc0Name = "Dc0"      # name of data compression sample
Dc0IsPolled = 1        # work mode, 0 means the "Interrupt" mode, 1 means the "Polling" mode
   2 means "Epoll Mode"
# List of core affinities
Dc0CoreAffinity = 1      # specifies CPU affinity for data compression
```

For more details on the configuration file, please visit Intel® QuickAssist Technology and refer to “Intel® QuickAssist Technology Software for Linux* - Programmer's Guide”.

6.3. Installing and Using QATzip

QATzip runs in the accelerated compression/decompression library in the user space. It is built on top of Intel QAT and offloads actual compression/decompression requests to Intel QAT devices. For more details please refer to the QATzip webpage.

For various user mode applications with compression enabled, QATzip enables the fast integration of hardware acceleration solutions with no need to call Intel QAT driver APIs in the application. This greatly shortens the development process which requires tinkering of hardware details such as hardware configuration, exception handling, memory preparation for hardware DMA, and other such operations. It also helps a lot to accelerate the evaluation and deployment of Intel QAT hardware optimized solutions.

Installation method

1) Set environment variables. Set **ICP_ROOT** as the QAT driver source code directory, and **QZ_ROOT** as the QATzip source code directory.

2) Enable Hugepages

```
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
rmmod usdm_drv
insmod $ICP_ROOT/build/usdm_drv.ko max_huge_pages=1024 max_huge_pages_per_process =16
```

3) Compile and install QATzip

```
cd $QZ_ROOT
./configure --with-ICP_ROOT=$ICP_ROOT
make clean
make all install
```

4) Update configuration file
QATzip provides a sample configuration file, which can be used with the QAT driver.

```
cp $QZ_ROOT/config_file/$YOUR_PLATFORM/$CONFIG_TYPE/*.conf /etc
```

5) Restart **qat_service**

```
service qat_service restart
```

QATzip provides the qzip program, which can be used for file compression and decompression. It can be used as follows:

```
qzip -k $your_input_file (add -h for help)

This compression and decompression util could support below options:
"  -A, --algorithm   set algorithm type, currently only support deflate",
"  -d, --decompress  decompress",
"  -f, --force       force overwrite of output file and compress links",
"  -h, --help        give this help",
"  -H, --huffmanhdr  set huffman header type",
"  -k, --keep        keep (don't delete) input files",
"  -V, --version     display version number",
"  -L, --level       set compression level",
"  -C, --chunksz     set chunk size",
"  -O, --output      set output header format(gzip|gzipext|7z)",
  " -r,          set max inflight request number",
  " -R,          set Recursive mode for decompressing a directory
               It only supports for gzip/gzipext format and
               decompression operation",
  " -o,          set output file name"
```

The "**run_perf_test.sh**" script can be used to test the compression and decompression performance of QATzip.

```
 cd $QZ_ROOT/test/performance_tests
 ./run_perf_test.sh
```

7. Conclusion

Intel provides different solutions in data compression, Intel ISA-L and Intel IPP are software solutions, if need higher performance using a smaller compression ratio, you can choose Intel ISA-L, if need a higher compression ratio at cost of some performance, you can choose Intel IPP. Intel QAT is a hardware-based solution that offloads data to hardware for processing, providing extremely high performance, you can choose the right solution according to different application scenarios.

8. Feedback

We value your feedback. If you have comments (positive or negative) on this guide or are seeking something that is not part of this guide, please reach out and let us know what you think.

Notices & 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.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

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 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.