NVM Express over TCP for Intel® Ethernet Products

Configuration Guide

Ethernet Products Group (EPG)

May 2021
# Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>May 7, 2021</td>
<td>Document title change only.</td>
</tr>
<tr>
<td>2.0¹</td>
<td>July 23, 2020</td>
<td>Initial public release.</td>
</tr>
</tbody>
</table>

¹. There are no previous publicly-available versions of this document.
NOTE: This page intentionally left blank.
1.0 Introduction

NVM Express* (NVMe*) drives are high-speed, low-latency, solid-state drives (SSDs), that connect over the server Peripheral Component Interconnect Express* (PCIe*) bus.

The development of these high-performance drives has spurred new innovation in storage over networking protocols, which takes full advantage of the drive capabilities in data center and cloud environments.

NVMe over TCP provides networked storage at a latency level close to locally-mounted storage through a re-architected storage protocol that combines the use of low-latency/high-efficiency fabric technologies such as Remote Direct Memory Access (RDMA) or Fibre Channel (FC) with these high-speed NVMe drives. Recent additions to the protocol have also allowed for the use of standard TCP for a more mainstream option to high-speed NVMe drive connectivity.

Intel supports NVMe over TCP on all Intel® Ethernet product lines. This guide focuses on the Intel® Ethernet 800 Series in the examples.

1.1 Purpose

This document is a reference guide for configuring NVMe over TCP on Linux* operating systems using Intel® Ethernet 800 Series devices.

1.2 Terminology

Table 1. Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIOS</td>
<td>Basic Input Output System</td>
</tr>
<tr>
<td>DUT</td>
<td>Device Under Test</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>NVM Express</td>
<td>Non-Volatile Memory Express*</td>
</tr>
<tr>
<td>NVMe</td>
<td></td>
</tr>
<tr>
<td>PCI Express*</td>
<td>Peripheral Component Interconnect Express*</td>
</tr>
<tr>
<td>PCIe</td>
<td></td>
</tr>
<tr>
<td>RDMA</td>
<td>Remote Direct Memory Access</td>
</tr>
<tr>
<td>RHEL</td>
<td>Red Hat* Enterprise Linux</td>
</tr>
<tr>
<td>SELinux</td>
<td>Security Enhanced Linux</td>
</tr>
<tr>
<td>SSD</td>
<td>Solid State Drive</td>
</tr>
</tbody>
</table>
2.0 Prerequisites

2.1 Hardware Prerequisites

Target server platform:
- Intel® Ethernet 800 Series
- 40+ GB RAM
- 1+ PCIe Gen3 SSD with NVMe high performance controller interface

Client server platform(s):
- Intel® Ethernet 800 Series
- 40+ GB RAM

2.2 Software Prerequisites

- Red Hat Enterprise Linux (RHEL) 7.x or similar (guide was tested with RHEL 7.6)
- Latest stable kernel: https://www.kernel.org (recommend 5.0.x or greater)
- configshell_fb (required to setup nvmetcli): https://github.com/openiscsi/configshell-fb/releases
- nvme-cli: https://github.com/linux-nvme/nvme-cli/releases
- fio: https://github.com/axboe/fio/releases
- Latest driver/NVM upgrades for the network interface under test (Intel® Ethernet 800 Series).
  Refer to the driver README files for driver-specific installation requirements and dependencies.
3.0 Upgrade Kernel (All Servers)

NVMe over TCP requires the systems to be on a recent stable kernel (5.0.x is required, 5.2+ is recommended) from kernel.org for NVMe over TCP’s latest patches/fixes.

3.1 Create Kernel .config File

1. Ensure that the ncurses-devel and openssl-devel packages are installed:
   ```
   yum install ncurses-devel
   yum install openssl-devel
   ```

2. Make the config file based on current settings:
   ```
   cd <path to kernel>
   make olddefconfig
   ```

3. Change the config file either manually or through make menuconfig to ensure the following options are set in the .config file:
   ```
   grep NVM .config
   CONFIG_NVME_CORE=m
   CONFIG_BLK_DEV_NVME=m
   CONFIG_BLK_DEV_NVME_SCSI=y
   CONFIG_NVME_FABRICS=m
   CONFIG_NVME_TARGET=m
   CONFIG_NVME_TARGET_TCP=m
   CONFIG_NVME_TCP=m
   ```

3.2 Build Kernel

1. Save the config file and build the OS:
   ```
   make -j 8
   make modules_install -j 8
   make install -j 8
   ```

2. Set updated kernel as default.

3. Reboot into the updated kernel.

3.3 BIOS Tunings

For best performance with NVMe over TCP, the following BIOS settings are recommended (the exact names might change according to the platform make/model):

- Disable hyper-threading (logical processors)
- Disable power management:
  - Set the power profile to Performance, enable Turbo, and disable the Cstates.
3.4 **(Optional) System TCP Tunings**

For best performance with NVMe over TCP, the following system tuning might be helpful.

**Note:** These settings are not persistent between reboots.

**Note:** System tuning is subjective and dependent on many factors in your environment, including platform specifications and software workload. Experimentation is recommended to find the best settings for a particular workload.

1. Disable firewalls.
   ```
   service firewalld stop; systemctl mask firewalld
   ```

   ```
   Change SELINUX=enforcing to SELINUX=disabled in /etc/selinux/config.
   ```

3. Set the CPU scaling governor to performance mode.
   ```
   x86_energy_perf_policy performance
   ```
4.0 Configure NIC Drivers/NVM (All Servers)

4.1 Install Intel LAN Driver

Download the latest Linux driver package from Intel for the Device Under Test (DUT) and follow the installation procedure outlined in the included README file to install the Linux driver.

4.2 Upgrade NVM to Latest Image on NIC

Download the latest NVM upgrade package from Intel for the NIC and follow the included documentation to perform the upgrade.

4.3 (Optional) Enable Jumbo Frames

For NVMe over TCP workloads, performance can be greatly improved by enabling 5K or 9K jumbo frames rather than standard MTU.

**Note:** The MTU size must be configured end-to-end on the network, including on any switch ports. For example, the following command sets MTU of 9K on the network interface specified by $iface.

```plaintext
ip link set $iface mtu 9000
```
5.0 Configure NVMe over TCP Target (Storage Server)

5.1 Install NVMe over TCP Tools

5.1.1 Install nvme-cli

1. Download nvme-cli from:
   https://github.com/linux-nvme/nvme-cli/releases

2. Install with the following command:
   
   ```
   cd /<path_to_nvme-cli>/; python setup.py install
   ```

5.1.2 Install nvmetcli

1. Download configshell_fb (required to setup nvmetcli) from:
   https://github.com/open-iscsi/configshell-fb/releases

2. Install with the following command:
   
   ```
   cd /<path_to_configshell-fb>/; python setup.py install
   ```

3. Download nvmetcli from:

4. Install with the following command:
   
   ```
   cd /<path_to_nvmetcli>/; python setup.py install
   ```

5.2 Configure NVMe Drives

5.2.1 Install Latest Drivers and Firmware for NVMe Drives

Follow NVMe drive manufacturer instructions.

5.2.2 Format NVMe Drives

Reformatting NVMe drives can be done using nvme-cli.

The following example code snippet formats all NVMe partitions on the server using nvme-cli:

```bash
for device in $(ls /dev/nvme*n*p*); do
    nvme format $device
done
```
5.2.3 Partition NVMe Drives

Partition the NVMe drives into as many partitions as needed using `gdisk`. Refer to the `gdisk` documentation for more details.

The following example script partitions all NVMe drives on the server into a specified number of equal partitions. This removes all data on the drives.

```
add_partitions_gdisk.sh
#!/bin/bash
partitions=2
for device in $(ls /dev/nvme*n* | grep -v p); do
    inputString=""
    echo "$device"
    max_sectorsize=$(echo -e "n\n" | gdisk $device |grep sector |grep -oP '(?<=34-).*?(?=,)')
    partNum=$((max_sectorsize/partitions))
    for (( i=1; i<=$partitions; i++))do
        partSize=$((partNum*i))
        echo $partSize
        inputString+="m\n\n$partSize\n\n"
    done
    inputString+="w\ny"
    printf $inputString | gdisk $device
done
```

5.3 Configure NVMe Target System

5.3.1 Load Modules

Load these modules before setting up the subsystems:

```
modprobe nvme nvnet null_blk nvmet_tcp
```

5.3.2 Configure NVMe Subsystems

**Note:** Refer to the `nvmetcli` documentation for the latest instructions.

There are two options for creating NVMe subsystems using `nvmetcli`:

- Use the `nvmetcli` interactive menu.
- Create the configuration file and use `nvmetcli` restore to load the file.

5.3.2.1 Option A: Create Subsystems Using nvmetcli Interactive Commands

The following example provides commands that interactively configure an NVMe subsystem with a single Logical Unit Number (LUN), where 4420 is the default port number for NVMe over TCP, and 10.10.10.20 is the IP Address for the target Ethernet interface:

```
nvmetcli
/> cd subsystems
/subsystems> create nvme4n1p1
/subsystems> cd nvme4n1p1/namespaces
/subsystems/nvme4n1p1/namespaces> create nsid=1
/subsystems/nvme4n1p1/namespaces> cd l
/subsystems/nvme4n1p1/namespaces/l> set device path=/dev/nvme4n1p1
Parameter path is now '/dev/nvme4n1p1'.
/subsystems/nvme4n1p1/namespaces/l> cd ..../
/subsystems/nvme4n1p1/namespaces> cd .....
/subsystems/nvme4n1p1> set attr allow_any_host=1
Parameter allow_any_host is now '1'.
/subsystems/nvme4n1p1> cd namespaces/1
```

Did this document help answer your questions?
/subsystems/n.../namespaces/1> enable
The Namespace has been enabled.
/subsystems/n.../namespaces/1> cd ..../..
/> cd ports
/> create 1
/> cd l
/> set addr adrfam=ipv4
Parameter adrfam is now 'ipv4'.
/> set addr trtype=tcp
Parameter trtype is now 'tcp'.
/> set addr trsvcid=4420
Parameter trsvcid is now '4420'.
/> set addr traddr=10.10.10.20
Parameter traddr is now '10.10.10.20'.
/> cd subsystems
/> subsystems> create nvme4n1p1

To save the target configuration to a file:

/> subsystems> saveconfig manual-config.json
/> subsystems> exit

5.3.2.2 Option B: Use nvmetcli to Load Saved Configuration

Create NVMe subsystems using nvmetcli restore [config file]:

nvmetcli restore savedconfig.json

Note: Refer to Section 8.0 for sample script to generate a saved config file automatically.

5.3.3 Clear NVMe Subsystems Using nvmetcli

To clear all NVMe subsystems:

nvmetcli clear
6.0 Configure NVMe over TCP Client(s)

6.1 Install NVMe over TCP Tools

6.1.1 Install nvme-cli

1. Download nvme-cli from:
   https://github.com/linux-nvme/nvme-cli/releases

2. Install with the following command:
   ```
cd /<path_to_nvme-cli>/; python setup.py install
   ```

6.2 Load Modules

Load these modules before setting up the subsystems:

```
modprobe configfs nvme nvme.tcp
```

6.3 Connect NVMe Drives

1. Discover NVMe drives available for connection:
   ```
   nvme discover -t tcp -a <targetIP> -s 4420
   ```

2. Connect the client to the target and mount an NVMe drive on the client:
   ```
   nvme connect -t tcp -s 4420 -a <targetIP> -n <target_disk_nqn>
   ```
   For example:
   ```
   nvme connect -t tcp -s 4420 -a 10.10.10.20 -n /dev/nvme2n1p1
   ```

6.4 Verify NVMe over TCP Connections

To verify drives are mounted, run the following commands:

```
nvme list
lsblk
```

**Note:** By default, regardless of the name or NQN of the subsystem on the target, the client-mounted subsystems are be named /dev/nvme[##]n1; where [##] is a number starting at 0 (or the lowest available if other NVMe drives are on the system) and incrementing as more NVMe drives are added/mounted.
7.0 Testing NVMe over TCP

Because NVMe over TCP is a block-based storage protocol, a standard block storage benchmark such as fio can be used to test performance.

7.1 Install fio

1. Download fio:
   https://github.com/axboe/fio/releases

2. Install with the following command:
   cd /<path_to_fio>/;./configure; make; make install

7.2 Precondition NVMe Drives

If performance testing on NAND-based NVMe drives, preconditioning the drives must be completed. If performance testing on Intel® Optane™ DC SSDs, no preconditioning is required.

For the best performance results, when using NAND-based NVMe drives, precondition the drives first using the fio tool.

To precondition the drives, fill up the entire span of the drive with sequential writes twice to ensure each of the Logical Block Addresses (LBAs) is written to.

**Note:** To benchmark 4 KB random writes, precondition the drives with 4 KB random writes instead of 1 MB sequential writes.

The following example fio command preconditions an NVMe drive (Sequential Writes: 1 MB Write, 32 IO depth, 1200 seconds for a 800 GB NVMe drive):

```
fio --filename=/dev/nvme0n1 --numjobs=1 --rw=write --iodepth=32 --bs=1M --runtime=1200
```

7.3 fio Example

**Note:** For the latest instructions, refer to the fio documentation included in the version being used.

Following is an example fio test (tests 4 KB read, 16 IO depth, 120 seconds, 4subsystems):

**Run on the client:**

```
fio fio.conf
```

**Where fio.conf contains:**

```
[global]
ioengine=libaio
rw=randread
iodepth=16
bs=4K
rwmixread=100
direct=1
time_based=1
runtime=120s
norandommap=1
numjobs=1

[filename1]
filename=/dev/nvme0n1
```
[filename2]
filename=/dev/nvme1n1
[filename3]
filename=/dev/nvme2n1
[filename4]
filename=/dev/nvme3n1
8.0 Example Script to Generate nvmetcli Subsystem Config File

Note: This script is provided as example only. Modify the script as needed to fit the environment.

gen_target_config_kernel_doc.sh
#This script creates nvmetcli configuration file that uses all nvme partitions in /dev/nvme*n*p* for target subsystem creation.
#Note: be sure to change $ip to match your DUT interface IP
#On NVMe target, after running this script, use 'nvmetcli restore <filen>' to create nvme subsystems

#file name for config file.
filen="nvme-target-setup-auto"

#DUT target interface IP
ip="10.10.10.20"

#get all NVMe drive partition paths
nvmePartitions=( $(ls /dev/nvme*n*p* | cut -d'/' -f3) )

#get last partition path in the list
lastnvmePartition=${nvmePartitions[$(( ${#nvmePartitions[*]} - 1 ))]}

#create config file
cat > $filen << EOF
{
  "hosts": [],
  "ports": [
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "$ip",
        "treq": "not specified",
        "trsvcid": "4420",
        "trtype": "tcp"
      },
      "portid": 1,
      "referrals": [],
      "subsystems": [
EOF

for dev in "${nvmePartitions[@]}"; do
  if [[ $dev == "$lastnvmePartition" ]]; then
    cat >> $filen << EOF
      "$dev"
EOF
  else
    cat >> $filen << EOF
      "$dev",
EOF
  fi
done

EOF

Did this document help answer your questions?
cat >> $filen << EOF
    
},
  "subsystems": [
EOF

i=1
for dev in "${nvmePartitions[@]}"; do
cat >> $filen << EOF
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1"
    },
    "namespaces": [
      {
        "device": {
          "nguid": "00000000-0000-0000-0000-000000000000",
          "path": "/dev/$dev"
        },
        "enable": 1,
        "nsid": 1
      }
    ],
    "nqn": "$dev"
EOF

if [[ $dev == "$lastnvmePartition" ]]; then
cat >> $filen << EOF
  ]
EOF
else
cat >> $filen << EOF
  }
EOF
fi
done
cat >> $filen << EOF
]}
EOF

Did this document help answer your questions?
No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

This document (and any related software) is Intel copyrighted material, and your use is governed by the express license under which it is provided to you. Unless the license provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this document (and related materials) without Intel's prior written permission. This document (and related materials) is provided as is, with no express or implied warranties, other than those that are expressly stated in the license.

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.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.