Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

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.

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 Intel product specifications and roadmaps.

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.

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, the Intel logo, and Xeon are trademarks of Intel Corporation in the United States and other countries.

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

Copyright © 2017, Intel Corporation. All Rights Reserved.
# Contents

1 Introduction .......................................................................................................................... 6
   1.1 Purpose .......................................................................................................................... 6
   1.2 Terminology .................................................................................................................. 6

2 Prerequisites ......................................................................................................................... 7
   2.1 Hardware Prerequisites ............................................................................................... 7
   2.2 Software Prerequisites ............................................................................................... 7

3 Upgrade Kernel (All Servers) ............................................................................................. 8
   3.1 Create kernel .config File ......................................................................................... 8
   3.2 Build Kernel ............................................................................................................... 8
   3.3 BIOS Tunings ............................................................................................................. 9

4 Configure and Test iWARP RDMA (All Servers) ................................................................. 10
   4.1 Download rdma-core and Install Dependencies ......................................................... 10
   4.2 Build and Install rdma-core .................................................................................... 10
   4.3 Disable SELinux* and Firewall ............................................................................. 11
   4.4 Load Modules .......................................................................................................... 12
   4.5 Check RDMA ........................................................................................................... 12

5 Configure NVMe* over Fabrics Target Server .................................................................. 13
   5.1 Install NVMe* over Fabrics Tools ............................................................................. 13
      5.1.1 Install nvme-cli .................................................................................................. 13
      5.1.2 Install nvmetcli ............................................................................................... 13
   5.2 Configure NVMe* Drives ......................................................................................... 14
      5.2.1 Partition NVMe* Drives ................................................................................. 14
      5.2.2 Format NVMe* Drives .................................................................................... 15
      5.2.3 Pre-condition NVMe* Drives ......................................................................... 15
   5.3 Load Modules .......................................................................................................... 15
   5.4 Configure NVMe* Subsystems ................................................................................ 15
      5.4.1 Create Subsystems Using nvmetcli Interactive Commands ......................... 16
      5.4.2 Use nvmetcli to Load Saved Configuration .................................................. 17
      5.4.3 Clear NVMe* Subsystems using nvmetcli ..................................................... 17

6 Configure NVMe* over Fabrics Client(s) ......................................................................... 18
   6.1 Install NVMe* over Fabric Tools ............................................................................. 18
      6.1.1 Install nvme-cli ................................................................................................ 18
   6.2 Load Modules .......................................................................................................... 18
   6.3 Connect NVMe* Drives ......................................................................................... 18
   6.4 Verify NVMe* over Fabrics Connections ................................................................ 19

7 Testing NVMe* over Fabrics ............................................................................................. 20
   7.1 fio Example .............................................................................................................. 20

A Appendix A — Script to Generate nvmetcli Subsystem Config File .............................. 21
Tables

Table 1-1. Terminology

6
### Revision History

<table>
<thead>
<tr>
<th>Document Number</th>
<th>Revision Number</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>336478-001</td>
<td>1.0</td>
<td>Initial release of the document.</td>
<td>September 2017</td>
</tr>
</tbody>
</table>
1 Introduction

NVM Express* (NVMe*) drives are high-speed, low-latency solid-state drives (SSDs), which 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 take full advantage of the drive capabilities in datacenter and cloud environments.

NVMe over Fabrics aims to provide 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) and Fibre Channel (FC) with these high-speed NVMe drives.

Intel® Ethernet Connection X722 supports NVMe over Fabrics with iWARP RDMA technology, offering a low latency/high bandwidth solution integrated into the Intel® C620 series chipset with the latest Intel® Xeon® Scalable processor generation platforms.

1.1 Purpose

This document is a reference guide for configuring NVMe over Fabrics on Linux* operating systems using Intel® Ethernet Connection X722 iWARP RDMA.

1.2 Terminology

Table 1-1. 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>FC</td>
<td>Fibre Channel</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>LUN</td>
<td>Logical Unit Number</td>
</tr>
<tr>
<td>NVMe*</td>
<td>Non-Volatile Memory Express</td>
</tr>
<tr>
<td>NVM Express*</td>
<td>Non-Volatile Memory Express* or NVM Express* or NVMe*</td>
</tr>
<tr>
<td>PCIe</td>
<td>Peripheral Component Interconnect Express*</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 drives</td>
</tr>
</tbody>
</table>

§
2 Prerequisites

2.1 Hardware Prerequisites

- Target server platform:
  - Intel® Xeon® Scalable processors
  - Intel® C620 series chipset with integrated Intel® Ethernet Connection X722
  - 16+ GB RAM
  - 1+ PCIe* Gen3 SSD with NVM Express* (NVMe*) high performance controller interface

- Client server platform(s):
  - Intel Xeon Scalable processors
  - Intel C620 series chipset with integrated Intel Ethernet Connection X722
  - 16+ GB RAM

2.2 Software Prerequisites

- Red Hat® Enterprise Linux® (RHEL) 7.x or similar (guide was tested with RHEL 7.1)
- Latest stable kernel: kernel.org (recommend 4.10 or greater)
- rdma_core: https://github.com/linux-rdma/rdma-core/releases
- nvmetcli: http://git.infradead.org/users/hch/nvmetcli.git
- configshell_fb (required to setup nvmetcli): https://github.com/open-iscsi/configshell-fb
- nvme-cli: https://github.com/linux-nvme/nvme-cli
- fio: https://github.com/axboe/fio/releases
Upgrade Kernel (All Servers)

NVMe* over Fabrics requires the systems to be on a recent stable kernel (4.10+ is recommended) from kernel.org for the NVMe over Fabrics latest patches/fixes. It is recommended to get the latest stable release from kernel.org, which should have the latest inbox driver (i40e and i40iw) for Intel® Ethernet Connection X722.

3.1 Create kernel .config File

**Note:** Ensure ncurses-devel and openssl-devel packages are installed

```bash
yum install ncurses-devel
yum install openssl-devel
```

Make config file based on current settings:

```bash
cd <path to kernel>
make olddefconfig
```

Change the config file either manually or through make menuconfig to ensure the following options are set in the .config file

```bash
grep NVM .config
CONFIG_NVME_CORE=m
CONFIG_BLK_DEV_NVME=m
CONFIG_BLK_DEV_NVME_SCSI=y
CONFIG_NVME_FABRICS=m
CONFIG_NVME_RDMA=m
CONFIG_NVME_TARGET=m
CONFIG_NVME_TARGET_LOOP=m
# CONFIG_NVM is not set
CONFIG_NVMEM=m

grep I40IW .config
CONFIG_INFINIBAND_I40IW=m
```

3.2 Build Kernel

Save the config file and build the OS:

```bash
make -j 8
make modules_install -j 8
make install -j 8
```

Reboot into the updated kernel.
3.3 BIOS Tunings

For best performance with NVMe over Fabrics, the following Basic Input Output System (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 C-states.

§
4 Configure and Test iWARP RDMA (All Servers)

4.1 Download rdma-core and Install Dependencies

rdma-core provides the necessary user space libraries to test Remote Direct Memory Access (RDMA) connectivity with tests such as rping.

Refer to the latest rdma-core documentation for updated installation guidelines. The following procedure is used to build rdma-core-14.

1. Download rdma_core from: https://github.com/linux-rdma/rdma-core/releases
2. Install dependencies as required.
   
yum install cmake gcc libudev-devel make pkgconfig libnl3
libnl3-devel valgrind valgrind-devel

Note: Specific versions of certain packages may be required.

Example: for rdma-core-14:

libnl3-3.2.28-2.el7.x86_64.rpm
valgrind-3.11.0-24.el7.x86_64.rpm
libnl3-cli-3.2.28-2.el7.x86_64.rpm
valgrind-devel-3.11.0-24.el7.x86_64.rpm
libnl3-devel-3.2.28-2.el7.x86_64.rpm

4.2 Build and Install rdma-core

1. Download the latest rdma-core*.tar.gz file from github.
2. Rename rdma-core*.tar.gz to rdma-core*.tgz
3. Untar rdma-core*.tgz
4. Create the /root/rpmbuild/SOURCES directory and copy the rdmac-core*.tgz file to it.
5. Build rpm:
   
rpmbuild -ba <path_to_rdma-core>/rdma-core.spec
6. Install rpm:
   
rpm -ivh /root/rpmbuild/RPMS/<rdma-core.rpm>
4.3 Disable SELinux* and Firewall

When running performance testing, disabling the firewall and Security-Enhanced Linux* (SELinux*) is recommended.

Disable the firewall:

```
systemctl stop firewalld
systemctl mask firewalld
```

Disable SELinux by editing the following file and changing enforcing to disabled:

```
vi /etc/selinux/conf
# This file controls the state of SELinux on the system.
# SELINUXTYPE.windows can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# targeted - No SELinux policy is loaded.
SELINUX=targeted
# SELINUXTYPE.windows can take one of three two values:
# targeted - Targeted processes are protected,
# minimum - Modification of targeted policy. Only selected
# processes are protected.
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```
4.4 Load Modules

Ensure the following modules are loaded:
```
modprobe ib_core i40iw iw_cm rdma_cm rdma_ucm ib_cm ib_uverbs
```

4.5 Check RDMA

Ensure that the RDMA interfaces listed on each server are shown when running the following command:
```
ibv_devices
```

Use `rping` to check for RDMA connectivity between target interface and client interface.

1. Assign IPs to the RDMA interfaces on Target and Client
2. On Target run:
   ```
   rping -sdVa <Target IP>
   ```
3. On Client run:
   ```
   rping -cdVa <Target IP>
   ```
   Press **Ctrl c** to exit rping.
5 Configure NVMe\* over Fabrics Target Server

5.1 Install NVMe\* over Fabrics Tools

5.1.1 Install nvme-cli

1. Download nvme-cli from:
   https://github.com/linux-nvme/nvme-cli
2. Install:
   cd /<path_to_nvme-cli>/; python setup.py install

5.1.2 Install nvmetcli

3. Download nvmetcli from:
   http://git.infradead.org/users/hch/nvmetcli.git
4. Install:
   cd /<path_to_nvmetcli>/; python setup.py install
5.2 **Configure NVMe\* Drives**

5.2.1 **Partition NVMe\* Drives**

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

*Note:* 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.**

```bash
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" | gdisk $device | grep sector | grep -oP '(?<=34-).*(?=,)')
  partNum=$((max_sectorsize/partitions))
  for (( i=1; i<= $partitions; i++))
  do
    partSize=$((partNum*i))
    echo $partSize
    inputString+="n\n\n\n$partSize\n\n"
  done
  inputString+="w\ny\n"
  printf $inputString | gdisk $device
done
```
5.2.2 **Format NVMe* Drives**

Reformatting NVMe drives can be done using `nvme-cli`.

The following 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 **Pre-condition NVMe* Drives**

If performance testing on NAND-based NVMe drives is to be completed, precondition the drives.

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

5.3 **Load Modules**

Load these modules before setting up the subsystems:

```bash
modprobe nvme nvmet null_blk nvmet_rdma
```

5.4 **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.
  
  —or—

- Create the configuration file and use `nvmetcli restore` to load the file.
5.4.1 **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 Fabrics, and 10.10.10.20 is the Internet Protocol (IP) address for the target Intel® Ethernet Connection X722 interface:

```
nvmetcli
/> cd subsystems
/subsystems> create nvme4nlp1
/subsystems> cd nvme4nlp1/namespaces
/subsystems/nvme4nlp1/namespaces> create nsid=1
/subsystems/nvme4nlp1/namespaces> cd 1
/subsystems/nvme4nlp1/namespaces/1> set device path=/dev/nvme4nlp1
Parameter path is now '/dev/nvme4nlp1'.
/subsystems/nvme4nlp1/namespaces/1> cd ../../../..
/> cd ports
/ports> create 1
/ports> cd 1
/ports/1> set addr adrfam=ipv4
Parameter adrfam is now 'ipv4'.
/ports/1> set addr trtype=rdma
Parameter trtype is now 'rdma'.
/ports/1> set addr trsvcid=4420
Parameter trsvcid is now '4420'.
/ports/1> set addr traddr=10.10.10.20
Parameter traddr is now '10.10.10.20'.
/ports/1> cd subsystems
/ports/1/subsystems> create nvme4nlp1
```

To save the target configuration to a file:

```
/ports/1/subsystems> saveconfig manual-config.json
/ports/1/subsystems> exit
```
5.4.2 **Use `nvmetcli` to Load Saved Configuration**

Create NVMe subsystems using `nvmetcli restore [config file]`

`nvmetcli restore savedconfig.json`

*Note:* Refer to Appendix A for sample script to generate a saved config file automatically.

5.4.3 **Clear NVMe* Subsystems using `nvmetcli`**

To clear all NVMe subsystems:

`nvmetcli clear`
6 Configure NVMe* over Fabrics Client(s)

6.1 Install NVMe* over Fabric Tools

6.1.1 Install nvme-cli

1. Download nvme-cli from:
   https://github.com/linux-nvme/nvme-cli
2. Install:
   cd /<path_to_nvme-cli>/; python setup.py install

6.2 Load Modules

Load these modules before setting up the subsystems:
   modprobe configfs
   modprobe nvme
   modprobe nvme_rdma

6.3 Connect NVMe* Drives

To discover NVMe* drives available for connection, run:
   nvme discover -t rdma -a <targetIP> -s 4420

To connect the client to target and mount an NVMe drive on client, run the following command:
   nvme connect -t rdma -s 4420 -a <targetIP> -n <target_disk_nqn>

Example:
   nvme connect -t rdma -s 4420 -a 10.10.10.20 -n /dev/nvme2np1
6.4 Verify NVMe* over Fabrics 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 Testing NVMe* over Fabrics

Because NVMe* over Fabric is a block-based storage protocol, a standard block storage benchmark such as fio (https://github.com/axboe/fio/releases) can be used to test performance.

7.1 fio Example

Note: Refer to fio documentation included the version being used for the latest instructions.

Example fio test (tests 4 kB read, 16 io depth, 120 seconds, 4 subsystems):

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
Appendix A — Script to Generate \texttt{nvmetcli} Subsystem Config File

\textbf{Note:}\quad This script is provided as example only. Modify the script as needed to fit the environment.

\texttt{gen_target_config_kernel_doc.sh}

\#This script creates \texttt{nvmetcli} configuration file that uses all \texttt{nvme} partitions in /dev/nvme*n*p* for target subsystem creation.

\#Note: be sure to change $ip to match your RDMA interface IP

\#On NVMe target, after running this script, use \texttt{`nvmetcli restore <filen>`} to create \texttt{nvme} subsystems

\#file name for config file.
\texttt{filen="nvme-target-setup-auto"}

\#RDMA target interface IP
\texttt{ip="10.10.10.20"}

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

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

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

\texttt{for dev in "$\{\texttt{nvmePartitions[@]}\}"; do  
if [[ $dev == "$\{lastnvmePartition\}" ]]; then  
cat >> $filen << EOF
EOF
else
\end{verbatim}

```bash

"${dev}" EOF
else
cat >> $filen << EOF
  "${dev})",
EOF
fi
done

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
Appendix A — Script to Generate nvmetcli Subsystem Config File

}]
}
EOF

§