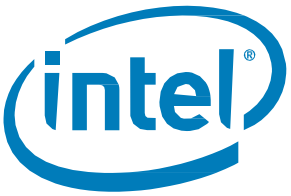

Intel® Linux* NVMe* Driver

Reference Guide for Developers

March 2015
Revision 2.0

Order Number: 330602-002US

Revision History

Revision	Description	Revision Date
001	Initial reference document.	June 2014
002	Updated for NVMe driver in kernel 3.19.	March 2015

Overview

The NVM Express* (NVMe*) is an optimized PCI Express* (PCIe*) solid-state drive interface. The NVMe specification defines an optimized register interface, command set, and feature set for PCIe-based SSDs.

For a background on NVMe, go to www.nvmexpress.org.

The NVM Express Linux driver development utilizes the typical open-source process used by kernel.org.

The development mailing list is linux-nvme@lists.infradead.org.

This document is intended for developer and software companies, it should be noted that kernel 3.3 had a stable NVMe driver version included, and various distributions have back ported the driver even to kernel 2.6 versions. The NVMe driver is also in-box with every current server distributions of Linux. Please check with your vendor. Intel encourages server user companies to focus on an in-box NVMe driver as your first option.

Getting Started

This section describes the required development tools and possible pre-requisites.

To clone, compile and build new kernel/driver, the following packages are required:

- **`ncurses`**
- **`build tools`**
- **`git`** (optional if using wget to get the Linux package)

NOTE: You must be root to install these packages

- Ubuntu based

```
apt-get install git-core build-essential libncurses5-dev
```

- RHEL based

```
yum install git-core ncurses ncurses-devel
yum groupinstall “Development Tools”
```

- SLES based

```
zypper install ncurses-devel git-core
zypper install --type pattern Basis-Devel
```

Building a New Linux Kernel with the NVMe Driver

Pick up a starting distribution, it does not matter from driver's perspective which distribution you use since it is going to put a new kernel on top of it, so use whatever you are most comfortable with and/or has the tools required.

The NVMe driver in kernel 3.19 integrates with new features in a way that makes it more serviceable and debug capable. We recommend this kernel if you are starting a project.

Kernel 3.19 brought a new block-mq model as the host of the NVMe driver. This added several reliability, availability, and serviceability features to the driver. The driver is now instrumented in a way that makes the debugging process much simpler, for example, `/sys/block/<device>/mq` has more statistics, and `blktraces` will be fully instrumented with events if needed.

1. Get the kernel and driver from the 3.x repository.

   Go to https://www.kernel.org/pub/linux/kernel/v3.x/

   For a snapshot, go to:

   wget https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.19.tar.xz

   ```
   tar –xvf linux-3.19.tar.xz
   ```

2. Build and install.

   a. Run **menuconfig** (which uses `ncurses`):

   ```
   make menuconfig
   ```

   b. Confirm that the NVMe Driver under Block is set to <M>

   Go to **Device Drivers-> Block Devices -> NVM Express block device**

   This creates the `.config` file in same directory.

   c. Run as root the following `make` commands (use the j flag as ½ your cores to improve make time)

   ```
   make –j10

   make modules_install install
   ```

   NOTE: Depending on distribution you use, you may have to run `update -initramfs` and `update -grub`, but this is typically unnecessary.

Once installation is successful, reboot the system to load the new kernel and drivers. Usually the new kernel becomes default to boot, which is the top line of `menu.lst`. This definition file is used for the GRUB booloader.

Verify the kernel version with "`uname –a`" after booting.

Use "`dmesg | grep –i error`" and resolve any kernel loading issues.

Running NVMe Driver Basic Tests

There are some basic open source NVMe test programs you can use for checking NVMe devices located at:

https://github.com/linux-nvme/nvme-cli

The typical installation steps are:

1. Git the source codes.

   ```
   git clone https://github.com/linux-nvme/nvme-cli
   ```

2. Make the testing programs.

   You can add or modify the Makefile with proper lib or header links and compile these programs:

   ```
   make
   $ sudo make install
   ```

3. Check the NVMe device controller “identify”, “namespace”, etc.

   For example:

   ```
   $ sudo nvme id-ctrl /dev/nvme0
   ```

Additional block counters can be found in `/sys/block/nvme*n*/mq/`.

Preparing the Drive

To zero out and condition a drive sequentially for performance testing, use the following command **twice**:

```
dd if=/dev/zero of=/dev/nvme0n1 bs=1M oflag=direct
```

We run this twice as a benchmarking function to guarantee that every block on the drive is erased, including the spare area.

You want to wait until the command reaches an EOF condition and ends.

NOTE: The data transferred by `dd` is not an indication of your application’s performance.

The preparation step is required if you are going to run some benchmarks on the drive. This puts the drive into sustained performance state, which is typically lower than the out of box performance.

We publish only sustained performance benchmarks in a drive datasheet to represent the real life performance. We recommend using sequential prefilling with the “dd” tool if you are going to test the sequential workloads. Use random 4k data filling if you will be testing random workloads.

Many users run a quick sanity test using `hdparm`, but this data should also not be used as an indication of reliable performance expectations.

```
hdparm -tT --direct /dev/nvme0n1
```

Aligning Drive Partitions

Be sure the starting block of the used partition is divisible by 4096 bytes. You can look at your partition table in the parted tool.

The following is an example listed partition table that is aligned:

```
Number  Start     End            Size           File system  Name     Flags
 1      1048576B  400088367103B  400087318528B               primary
```

Using the Start partition block value in bytes, you divide this number by 4096, like this:

```
1048576/4096 = 256
```

This shows an evenly divisible partition and so the partition will perform well.

Using the "`unit b`" option in parted will present partition start and end values in bytes.

Filesystem Recommendations

IMPORTANT: Do not discard blocks in filesystem usage.

Be sure to turn off the discard option when making your Linux filesystem. You want to allow the SSD manage blocks and its activity between the NVM (non-volatile memory) and host with more advanced and consistent approaches in the SSD Controller.

Core Filesystems:

- `ext4` – the default extended option is not to discard blocks at filesystem make time, retain this, and do not add the "discard" extended option as some information will tell you to do.
- `xfs` – with `mkfs.xfs`, add the `-K` option so that you do not discard blocks.

If you are going to use a software RAID, it is recommended to use a chunk size of 128k as starting point, depending on the workload you are going to run. You must always test your workload.