Past and present of the Linux NVMe driver

Christoph Hellwig
A driver..

**Definition of DRIVER**

: one that **drives:** such as
  a: coachman
  b: the operator of a motor vehicle
  c: an implement (such as a hammer) for **driving**
  d: a mechanical piece for imparting motion to another piece
  e: one that provides impulse or motivation • a **driver** in this economy
  f: a golf wood with a nearly straight face used in driving
  g: an electronic circuit that supplies input to another electronic circuit; **also**:
  h: a piece of computer software that controls input and output operations

(from https://www.merriam-webster.com/dictionary/driver)
Two drivers:

<table>
<thead>
<tr>
<th></th>
<th>virtio_scsi</th>
<th>lpfc</th>
</tr>
</thead>
<tbody>
<tr>
<td>.c files</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>.h files</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>LOC</td>
<td>1305</td>
<td>101,369</td>
</tr>
</tbody>
</table>

- Drivers can have orders of magnitude difference sizes
  - Types of supported hardware
  - Functionality
### Three (NVMe) drivers.

<table>
<thead>
<tr>
<th></th>
<th>Linux 4.4</th>
<th>Linux 4.12</th>
<th>OFED 1.5 (Win)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.c files</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>.h files</td>
<td>3</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>LOC</td>
<td>7501</td>
<td>2350</td>
<td>29689</td>
</tr>
</tbody>
</table>
Four drivers..

<table>
<thead>
<tr>
<th></th>
<th>Linux 4.4</th>
<th>Linux 4.12 (nvme)</th>
<th>Linux 4.12 (nvme-core)</th>
<th>OFED 1.5 (Win)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.c files</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>.h files</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>LOC</td>
<td>7501</td>
<td>2350</td>
<td>6525</td>
<td>29689</td>
</tr>
</tbody>
</table>
The humble beginning

author Matthew Wilcox <matthew.r.wilcox@intel.com> 2011-01-20 12:50:14 -0500
committer Matthew Wilcox <matthew.r.wilcox@intel.com> 2011-11-04 15:52:51 -0400
commit b68503ba432b16f8442a8d4e29a7aad2c0c363d (patch)
tree 43dca7cd57965ce1a2b7b6f94437f09364fbc0034
parent 8b934ccc787ff3a87f15a35a9916d1d8e85d30e (diff)
download linux-b68503ba432b16fc8442a84e29a7aad2c0c363d.tar.gz

**NVMe: New driver**

This driver is for devices that follow the NVM Express standard

Signed-off-by: Matthew Wilcox <matthew.r.wilcox@intel.com>

**Diffstat**

```plain
-rw-r-- Documentation/ioct1/ioct1-number.txt 1
-rw-r-- drivers/block/Kconfig 11
-rw-r-- drivers/block/Makefile 1
-rw-r-- drivers/block/nvme.c 1043
-rw-r-- include/linux/nvme.h 343
```

5 files changed, 1399 insertions, 0 deletions
The humble beginning

More than 1 month before the release of NVMe 1.0

NVMe: New driver
This driver is for devices that follow the NVM Express standard
Signed-off-by: Matthew Wilcox <mattwhwilcox@intel.com>

Diffstat

<table>
<thead>
<tr>
<th>Mode</th>
<th>File Path</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rw-r--</td>
<td>Documentation/ioct/Ioctl-Number.txt</td>
<td>1</td>
</tr>
<tr>
<td>-rw-r--</td>
<td>drivers/block/Kconfig</td>
<td>11</td>
</tr>
<tr>
<td>-rw-r--</td>
<td>drivers/block/Makefile</td>
<td>1</td>
</tr>
<tr>
<td>-rw-r--</td>
<td>drivers/block/nvme.c</td>
<td>1043</td>
</tr>
<tr>
<td>-rw-r--</td>
<td>include/linux/nvme.h</td>
<td>343</td>
</tr>
</tbody>
</table>

5 files changed, 1399 insertions, 0 deletions
Early days

First version (Jan 2011) was very limited:

- Single SQ/CQ only
- Small data transfers (PRP1 only)
- Read and Write I/O commands and a few admin commands

Improved version merged into Linux 3.3 (Jan 2012):

- Support for multiple queues
- Large data transfers using PRP chains
- Lots of fixes
- Drivers has grown by about 800 LOC
Junior years

Nothing too exciting until Nov 2015:

- Lots of bug fixes
- Support for deallocate ("discard")
- Actually working flush support
- `/dev/nvmeX` character devices
- Addition of a SCSI translation for ioctls
Using blk-mq in the NVMe driver

Linux 3.19 switch the NVMe driver to use blk-mq

- Allowed to remove hundreds of lines of code from the NVMe driver
- Very few modifications to the core blk-mq code were required
  - Most of that had been take care of for SCSI
- Building block for many future features
Blk-mq overview

What does blk-mq do?

- Split and merge I/O requests
- Manage multiple submission and completion queues
- Provide a command ID (tag) allocator
- Manage per-I/O data structures
- And much more

A bit of history:

- First prototyped in 2011
- Merged in Linux 3.13 (2014) for virtio
- Used by SCSI since 3.17 (2014)
- Used by NVMe since 3.19 (2015)
- And about a dozen other drivers now
PRPs and SGLs

PRPs describe each page:
- Offset 0
- Offset 4096
- Offset 8192

SGLs describe regions:
- Offset 0
  - Length 8192
- Offset 8192
  - Length 4096
PRPs and Linux

The Linux I/O stack uses SGL-like structures as they are more flexible:

- They store large contiguous regions efficiently
- Allow arbitrary borders between segments
- But we could also support PRPs relatively easily
  - Also needed for RDMA, Hyper-V

block: add support for limiting gaps in SG lists

Another restriction inherited for NVMe - those devices don’t support SG lists that have “gaps” in them. Gaps refers to cases where the previous SG entry doesn’t end on a page boundary. For NVMe, all SG entries must start at offset 0 (except the first) and end on a page boundary (except the last).

Signed-off-by: Jens Axboe <axboe@fb.com>
NVMe and SGLs

Since version 1.1 NVMe has optional SGL support

- Useful for large contiguous transfers, but use PRPs otherwise
- Can not be used for admin commands
- Except for NVMe over Fabrics, where only SGLs can be used

Linux support for SGLs is pending

- Patches are out on the mailing list
- Need better detection of contiguous regions
- ~ 5% performance benefit for large transfers
Coming of age

Lots of feature work after the blk-mq switch:

• T10 PI support (Feb 2015)
• CMB support, SQs only for now (Jul 2015)
• Persistent reservation support (Oct 2015)
• Support for weird Apple devices (Nov 2015)
• Basic SR-IOV support (Jun 2016)
From driver to subsystem, part 1

- Blk-mq allows passthrough requests that contain drivers specific raw commands
  - Initially used for SCSI CDBs
  - Generalized for NVMe commands
- Allowed us to split core vs PCIe to prepare for Fabrics
- Also supports multiple I/O command sets (e.g. LightNVM)
From driver to subsystem, part 2

Modularization for Fabrics:

- Move from `drivers/block/` to `drivers/nvme/host/` to prepare for a lot more NVMe related source files
- Split of the `nvme-core` module out of the existing `nvme` module also at the binary level
- Addition of new `nvme-rdma` and `nvme-fabrics` modules after the NVMe over Fabrics spec went public in June 2016
The NVMe (driver) subsystem

- 1 core NVMe module
  - Includes NVM I/O command set support
  - Including two optional features:
    - SCSI translation
    - LightNVM command set (Open Channel)
- NVMeOF library
- 3 transport drivers (PCIe, RDMA, FC)
Growing the family

• NVMe over Fabrics (RDMA) support (July 2016)
  – Including support for software defined NVMeOF controllers “target”

• Fibre Channel support (Dev 2016)
The chastity belt

TCG Opal support in Linux 4.11:

- Disk encryption and access control
- Generic library
- Less than 50 lines of code in NVMe
- Now also supports for ATA
I/O flow – traditional IRQ path

System call (read, write, ...)

User Process context

VFS + BIO stack

Device driver command submission

Wait (take a nap)

VFS + BIO stack

Other context

IRQ handler (device driver)

Hardware

Command execution
Hitting it hard

Linux 4.4 introduced a polled I/O mode

- Controlled by new RWF_HIPRI flag to the new preadv2/pwritev2 system calls
- Polling starts after I/O submission
- 100% CPU usage
Going hybrid

Linux 4.10 added hybrid polling

- Don’t start polling after submission – wait for half the average completion time
- Needs a good time estimate (especially for different I/O sizes)
- Still waists a lot of CPU – new patches to only start polling at the expected completion time
Polling latency
Coming of age

Catching up:

- Ranged deallocate support (Feb 2017)
- Autonomous Power State Transitions (Feb 2017)
- Host Memory Buffer support (May 2017)

And leading the pack with new NVMe 1.3 features:

- Set Doorbell Buffer, aka paravirtualized NVMe (Apr 2017)
- UUID identifiers (Jun 2017)
- Hot/Cold separation by (ab)using streams (Jun 2017)
Multipathing in NVMe

NVMe 1.1+ supports multiple controllers per subsystem

- Can be used to access shared storage from multiple systems
- Or to access data from the same system through different paths

Use cases for multi-path access

- Aggregate bandwidth over multiple connections
- Redundancy
- Locality of access
All roads lead to Rome.. (from www.aaroads.com)
Asynchronous Namespace Access

- Allows NVMe controllers to report access status per (namespace, controller) tuple
- Logical equivalent of ALUA in SCSI
(SCSI) Multipathing in Linux

- File system
- Device mapper multipath
- SCSI layer
- multipathd
Plan for NVMe Multipathering in Linux

Get the middle man out of loop

- NVMe already manages discovery of namespaces, and reporting ANA states
- Allows for automatic discovery and set up
- Allows for no added latency in NVMe vs additional 5-6 microseconds with device mapper
References

I/O Latency Optimization with Polling


Improving Block Discard Support throughout the Linux Storage Stack


Increasing SCSI LLD Driver Performance by using the SCSI Multimqueue Approach

Question?