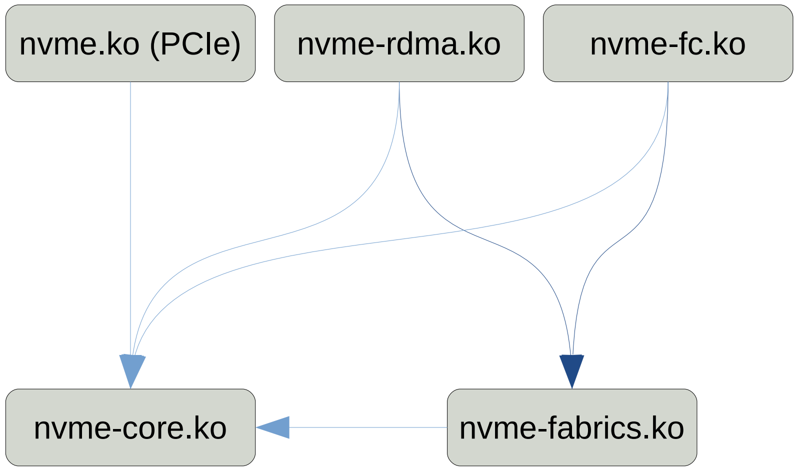
SDC19 | September 23-26, 2019
Santa Clara, CA

Linux NVMe and block layer status update

Christoph Hellwig

- Covers new material since my “*Past and present of the Linux NVMe driver*” talk from SDC 2017
- The latest kernel version in September 2017 was Linux 4.13, the latest kernel today is Linux 5.3.

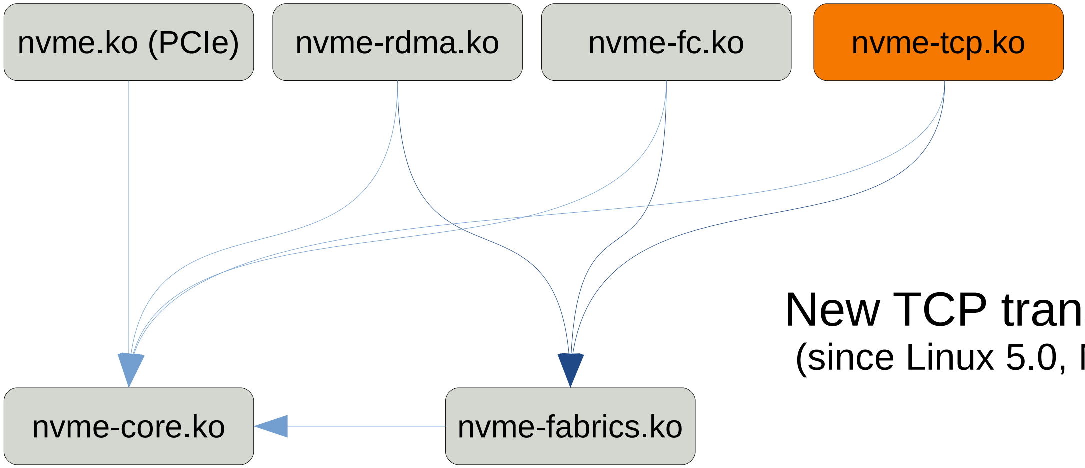
NVMe driver structure



Major new code modules

- TCP transport
- Core NVMe code:
  - Multipathing (including ANA)
  - NVMe tracing

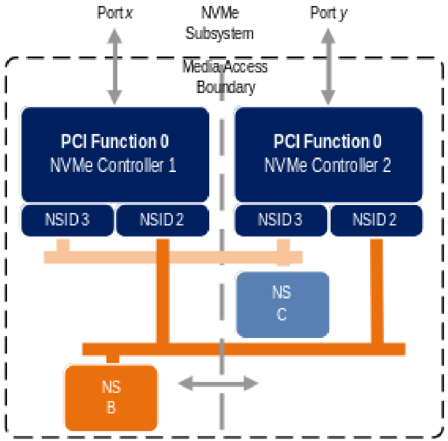
NVMe driver structure



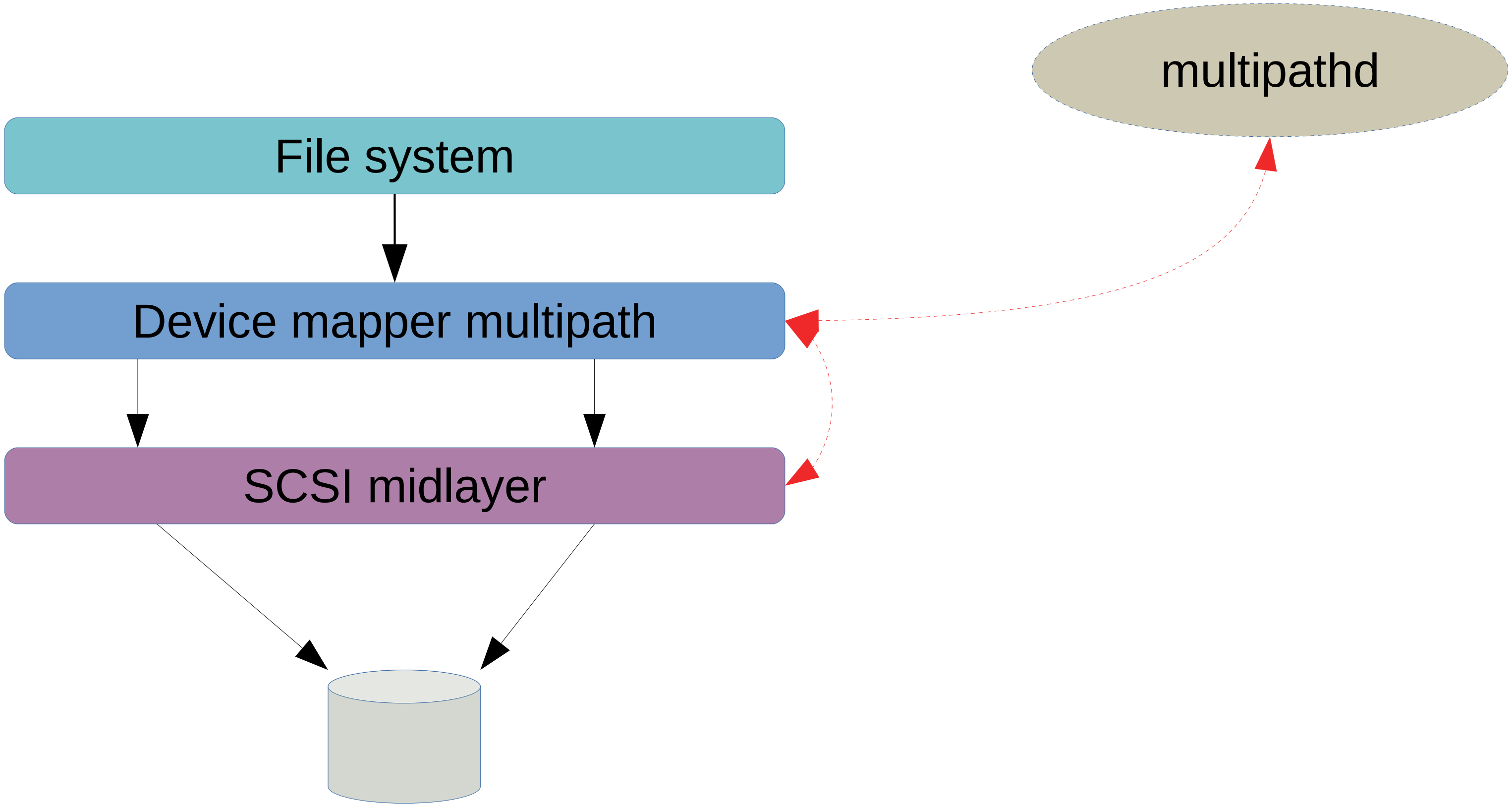
New TCP transport
(since Linux 5.0, Mar 2018)

Multipathing

- Multiple NVMe controllers can access the same namespace
  - Much tighter architecture model than SCSI
- Asynchronous Namespace Access (ANA) helps communicating access rules
  - Similar to ALUA in SCSI but simpler and more consistent
- Latency and IOPS matter and not just throughput!



Legacy SCSI multipathing



Native NVMe multipathing

- Small addition to the core NVMe driver (< 1k LOC including ANA)
  - Multiplexes access to the **/dev/nvmeXnY** block devices to multiple controllers if present, transparent to the file system / application.
  - Pathing decisions based on ANA state, NUMA proximity or optionally a simple round robin algorithm
  - Up to 6x better IOPS than dm-multipath while using less CPU cycles
  - Automatic setup

NVMe tracing

- Tracing of low-level NVMe command and queue state
  - **Does not replace blktrace!**

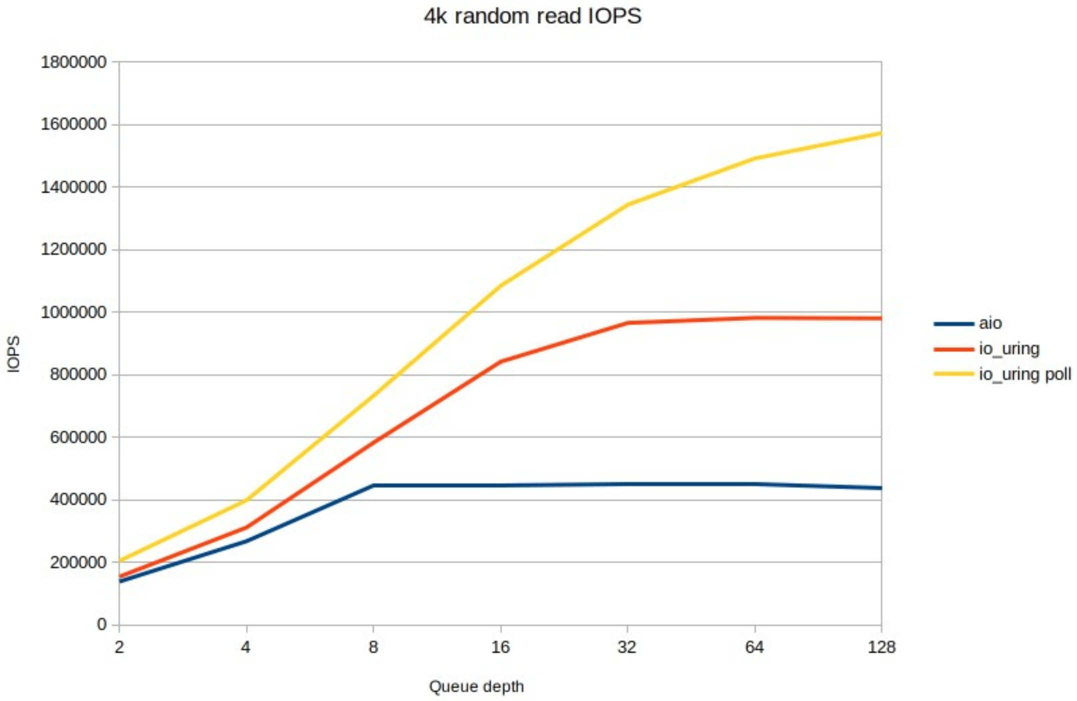
NVMe tracing output

```
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
   kworker/u16:2-9894  [000] .... 103979.005689: nvme_setup_cmd: nvme0: disk=nvme0n1, qid=1, cmdid=461, nsid=1,
flags=0x0, meta=0x0, cmd=(nvme_cmd_write slba=250534048, len=31, ctrl=0x0, dsmgmt=0, reftag=0)
   kworker/u16:2-9894  [000] .... 103979.005708: nvme_setup_cmd: nvme0: disk=nvme0n1, qid=1, cmdid=462, nsid=1,
flags=0x0, meta=0x0, cmd=(nvme_cmd_write slba=59542456, len=15, ctrl=0x0, dsmgmt=0, reftag=0)
   kworker/u16:2-9894  [000] .... 103979.005712: nvme_setup_cmd: nvme0: disk=nvme0n1, qid=1, cmdid=463, nsid=1,
flags=0x0, meta=0x0, cmd=(nvme_cmd_write slba=751153120, len=7, ctrl=0x0, dsmgmt=0, reftag=0)
          <idle>-0     [003] d.h. 103989.671361: nvme_complete_rq: nvme0: disk=nvme0n1, qid=4, cmdid=952, res=0,
retries=0, flags=0x0, status=0
          <idle>-0     [003] d.h. 103989.671392: nvme_complete_rq: nvme0: disk=nvme0n1, qid=4, cmdid=953, res=0,
retries=0, flags=0x0, status=0
```

Polling rework

- Polling NVMe completions have been supported since Linux 4.4
  - Polling was performed on the “normal” CQs by the submitting thread
  - Thus limited to QD=1
  - Hybrid polling (added in Linux 4.10) helped to drastically reduce CPU usage
- A new I/O approach has been developed to allow for batched polling (landed in Linux 5.1)
  - See the “***Improved Storage Performance Using the New Linux Kernel I/O Interface***” talk on Thursday for interface details!
  - Uses a dedicated polling CQ and a dedicated polling thread to perform millions of IOPs while using a single core
  - Initially supported for PCIe transport, now also on RDMA and TCP

io_uring performance



Graph from Jens Axboe via twitter

Scatter/Gather list support

- SGLs are the traditional way to specify data transfers in storage controllers
  - NVMe 1.0 only supported a different scheme called PRPs
  - Later SGL support was added to NVMe (PCIe)
  - Fabrics always used a SGL scheme
- Linux gained support for SGL on NVMe/PCIe in Linux 4.15 (Jan 2018)
  - Allow for much more efficient large data transfers

PRPs vs SGLs

PRP transfer:

Addr X

Addr X + 4096

Addr X + 8192

...

Addr X + 61440

SGL transfer:

Offset 0	Length 65536

Multipage bio_vec structures

```
struct bio_vec {
        struct page     *bv_page;
        unsigned int    bv_len;
        unsigned int    bv_offset;
};
```

- The bio_vec structure is a in-memory scatter/gather structure
  - Used everywhere in the block layer (and now also in the network stack)
  - Historically only used for fragments inside a page
  - Since Linux 5.0 (Mar 2019) can store arbitrarily large segments
- Together with SGL support allows transferring huge pages very efficiently

Single Segment optimizations

- Linux creates a “scatterlist” structure from the bio_vecs before submitting I/O
  - Helps with IOMMU batch mapping
  - Used to help with merging multiple bio_vecs
- The structure duplicates the bio_vecs and protocol-specific SGLs
  - Preferably avoid it entirely
  - For now we can only easily skip it for the single PRP/SGL entry case
- Up to 4% speedups for high-IOPS workloads (Linux 5.2)
  - Every cacheline counts!

Performance optimizations

- Dedicated read queues (Linux 5.1 for PCIe)
  - Allows placing reads on separate NVMe queues from other I/O
  - Later also added to RDMA/TCP
- Lockless CQs (Linux 5.1 for PCIe)
  - The addition of explicit poll queues allows lockless access to other CQs
- Batched doorbells writes (Linux 5.0 for PCIe)
  - Avoid ringing the SQ doorbell if more commands are pending
- Multiple inline segments (Linux 4.19 for RDMA)
  - Reduces protocol round trips and thus latency

PCIe Peer to Peer transfers

- Allow one PCIe device to transfer data from and to another without involving the host CPU
  - One device needs to present a memory BAR, and the other one accesses it
  - PCIe NVMe controllers can expose the Controller Memory Buffer (CMB) for P2P transfers.
  - The NVMe over Fabrics target can initiate P2P transfers from the RDMA HCA to / from the CMB
  - The PCI layer P2P framework, NVMe and RDMA support was added in Linux 4.19, still under development (e.g. IOMMU support)
- Warning: NVMe CMB support has grave bugs in virtualized environments!

Consumer grade NVMe

- NVMe has fully arrived in the consumer space
  - m.2 NVMe devices are everywhere, various BGA (solder on devices) as well, other new small form factors
  - A lot more buggy devices (up to 13 quirk bits now)
  - Linux 5.4 will support recent Apple Mac Book “NVMe” controllers
    - 128 byte SQEs, shared tags, single interrupt vectors

Power Management

- Linux uses Autonomous Power state transitions (APST) since Linux 4.12
  - Major runtime power savings
  - Lots of device / platform issues unfortunately
- Since Linux 5.3 Linux can also use APST for system suspend
  - Based on the Microsoft modern standby concept
  - Keeps causing problems with various device / platform combinations

Intel chipset problems

- Intel consumer chipset may run in “RAID” mode
  - Hides one or multiple NVMe controllers behind an AHCI controller
  - Not documented
  - Not easily discoverable
  - No way to quirk devices based on PCI IDs
  - No way to support SR-IOV or proper reset behavior
- Seems like an intentional sabotage by Intel
  - Causes major sabotage with Linux laptop support
  - Can’t be used by anything but the Intel binary windows driver blob

Questions?