Introduction to libnvme

Keith Busch

Linux Vault 2020

Forward-Looking Statement

Safe Harbor | Disclaimers

This presentation contains forward-looking statements that involve risks and uncertainties, including, but not limited to, statements regarding our technology research, market positioning for archival storage technology, growth opportunities, and market trends. Forward-looking statements should not be read as a guarantee of future performance or results, and will not necessarily be accurate indications of the times at, or by, which such performance or results will be achieved, if at all. Forward-looking statements are subject to risks and uncertainties that could cause actual performance or results to differ materially from those expressed in or suggested by the forward-looking statements.

Key risks and uncertainties include, among others: volatility in global economic conditions; development and introduction of products based on new technologies; unexpected advances in competing technologies; business conditions; and changes in data center demands and configurations. More information about the risks and uncertainties that could affect our business are listed in our filings with the Securities and Exchange Commission (the "SEC") and available on the SEC's website at www.sec.gov, including our most recently filed periodic report, to which your attention is directed. We do not undertake any obligation to publicly update or revise any forward-looking statement, whether as a result of new information, future developments or otherwise, except as required by law.

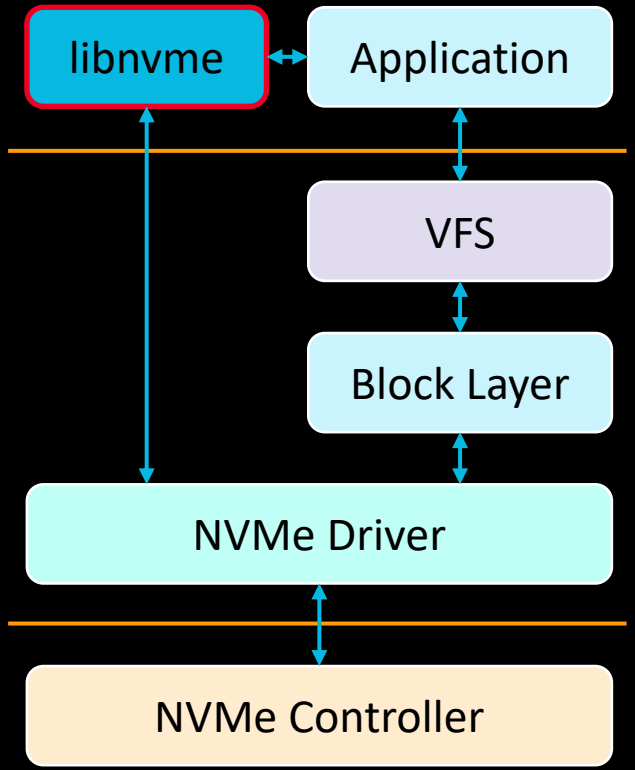
All trademarks are the property of their respective.

NVMe and Linux: Let's Make a Userspace Library

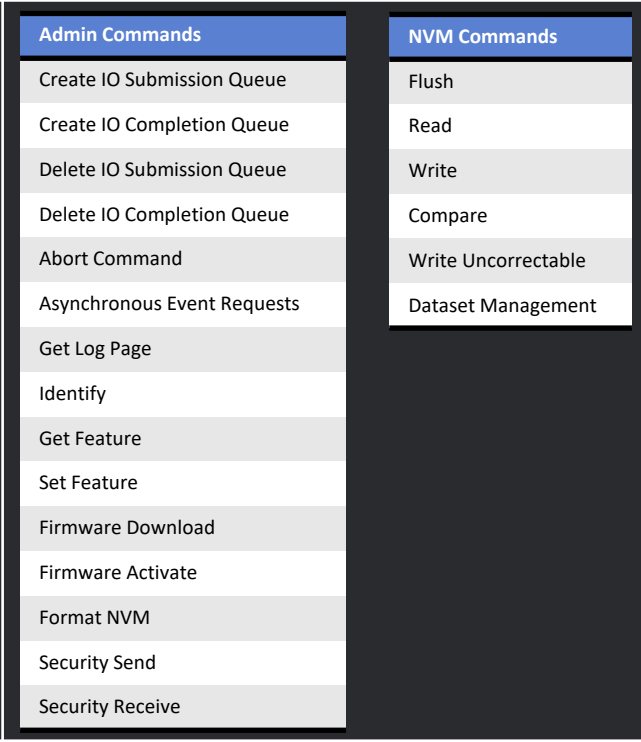
- NVM Express protocol and Linux features supporting it have grown significantly
- Open source and standard specification allows common software compatibility across a wide spectrum of devices
  - Host device drivers converge on common sources
  - But NVMe user space software often reimplements many of the same protocol'foundational core
- The nvme kernel user APIs are powerful, but too simple and abstract for easy use
- **libnvme**: Make all NVMe on Linux features conveniently reachable to developers

libnvme: Where it fits

- A place for all NVMe on Linux features: 0-day updates for ratified features and kernel driver changes
- Work with NVMe drivers (not around them)
  - The driver owns controller initialization, error recovery, transport specifics, command tagging, queuing and completion
  - libnvme provides methods to scan devices, connect to new targets, dispatch arbitrary commands to them and setup or decode data payloads



NVMe 1.0: The Early Days, circa 2011

Specifications	Transports	Commands
		

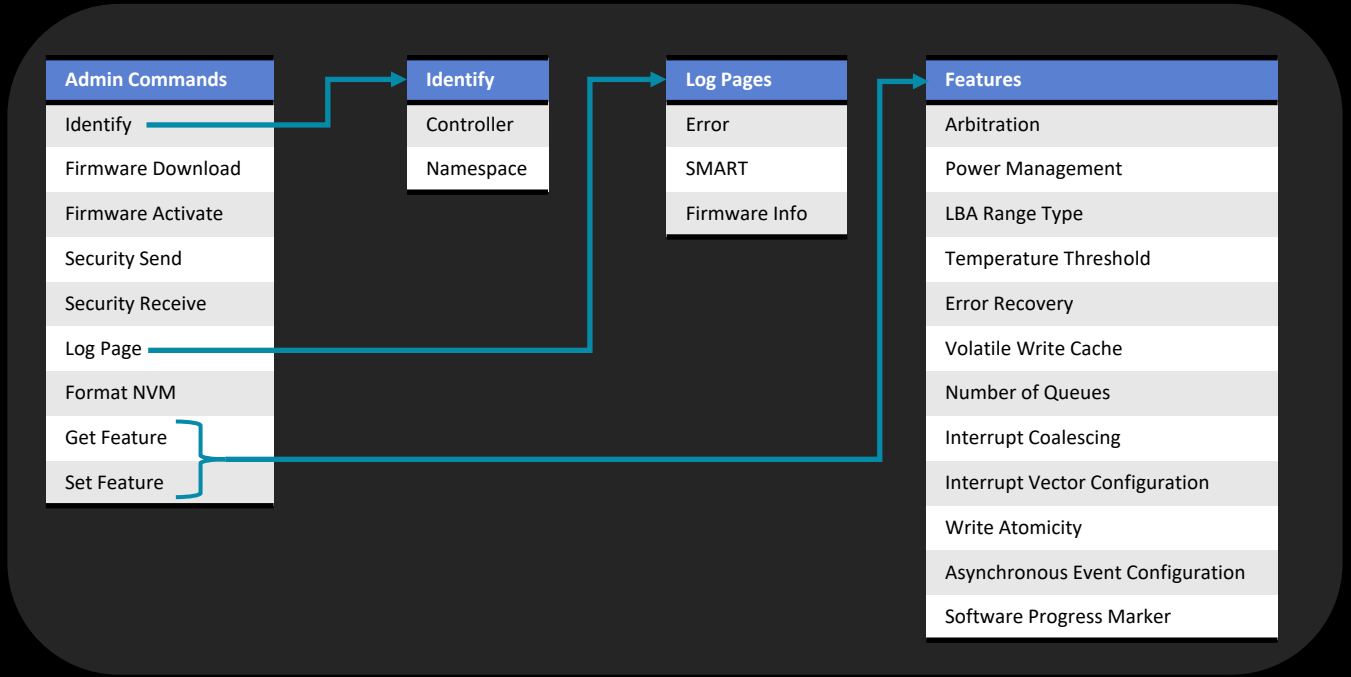
libnvme doesn't support harmful commands

Specifications	Transports	Commands

Admin Commands
~~Create IO Submission Queue~~
~~Create IO Completion Queue~~
~~Delete IO Submission Queue~~
~~Delete IO Completion Queue~~
~~Abort Command~~
~~Asynchronous Event Requests~~
Get Log Page
Identify
Get Feature
Set Feature
Firmware Download
Firmware Activate
Format NVM
Security Send
Security Receive

NVM Commands
Flush
Read
Write
Compare
Write Uncorrectable
Dataset Management

A few commands have subtypes



NVMe 1.4: Today

Specifications

Transports

PCI EXPRESS®

RDMA

Remote Direct Memory Access

FIBRE CHANNEL

TCP

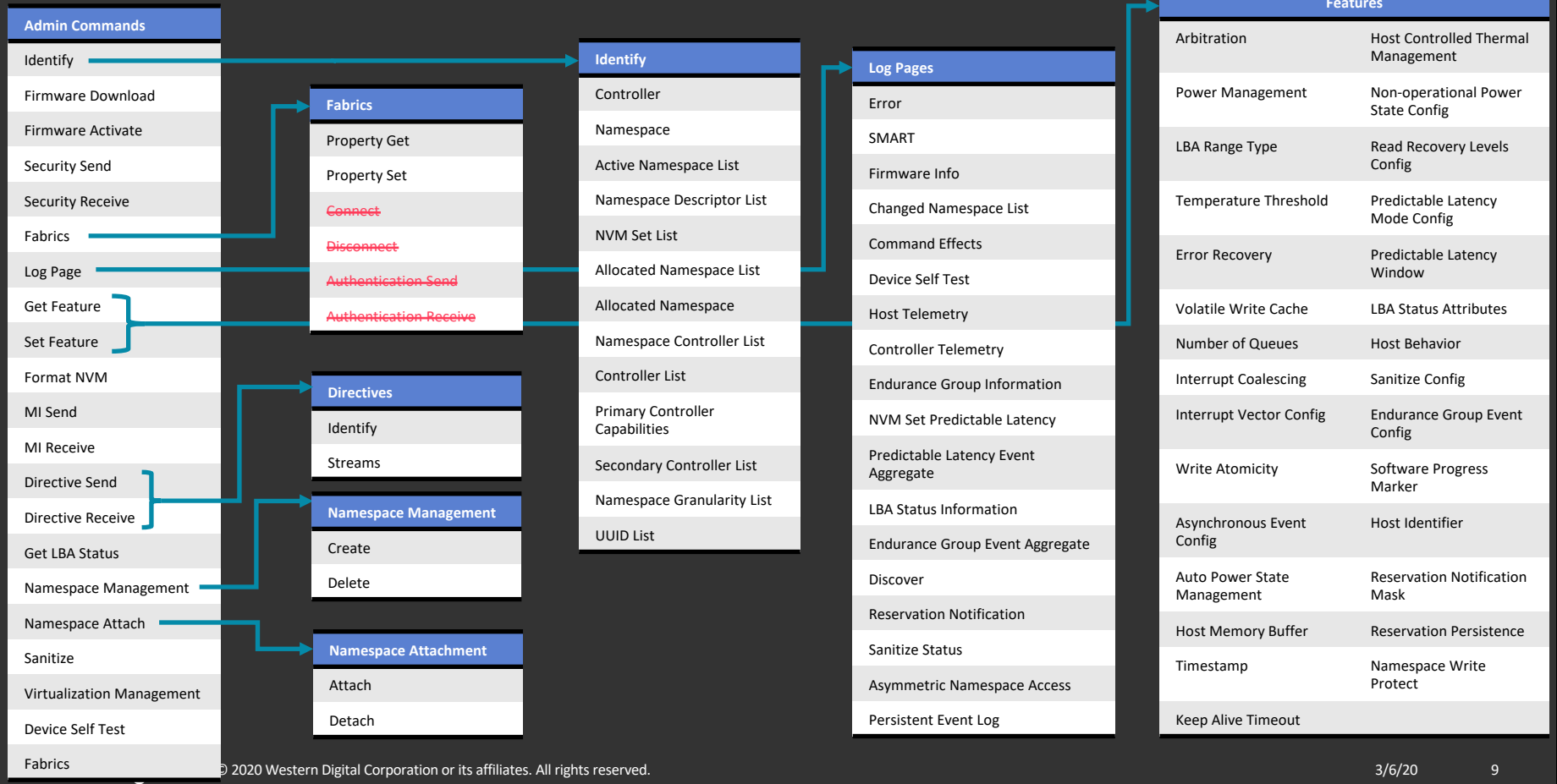
Other/Loop-back

Commands

Admin Commands	
Identify	Directive Send
Firmware Download	Directive Receive
Firmware ~~Activate~~Commit	Get LBA Status
Security Send	Namespace Management
Security Receive	Namespace Attach
Log Page	Sanitize
Format NVM	Virtualization Management
Get Feature	Device Self Test
Set Feature	Fabrics
MI Send	~~Keep Alive~~
MI Receive	

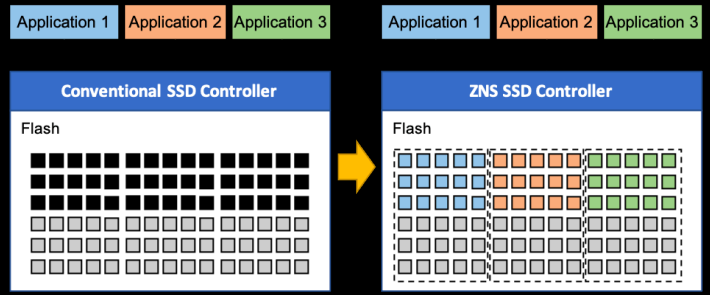
IO Commands
Flush
Read
Write
Compare
Write Uncorrectable
Dataset Management
Write Zeroes
Verify
Reservation Register
Reservation Acquire
Reservation Release
Reservation Report

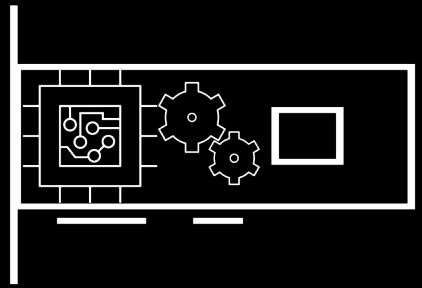
Many more command subtypes!

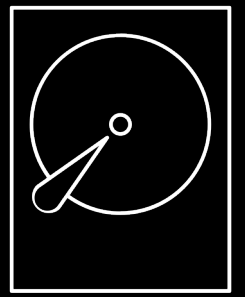


NVMe: New features under development

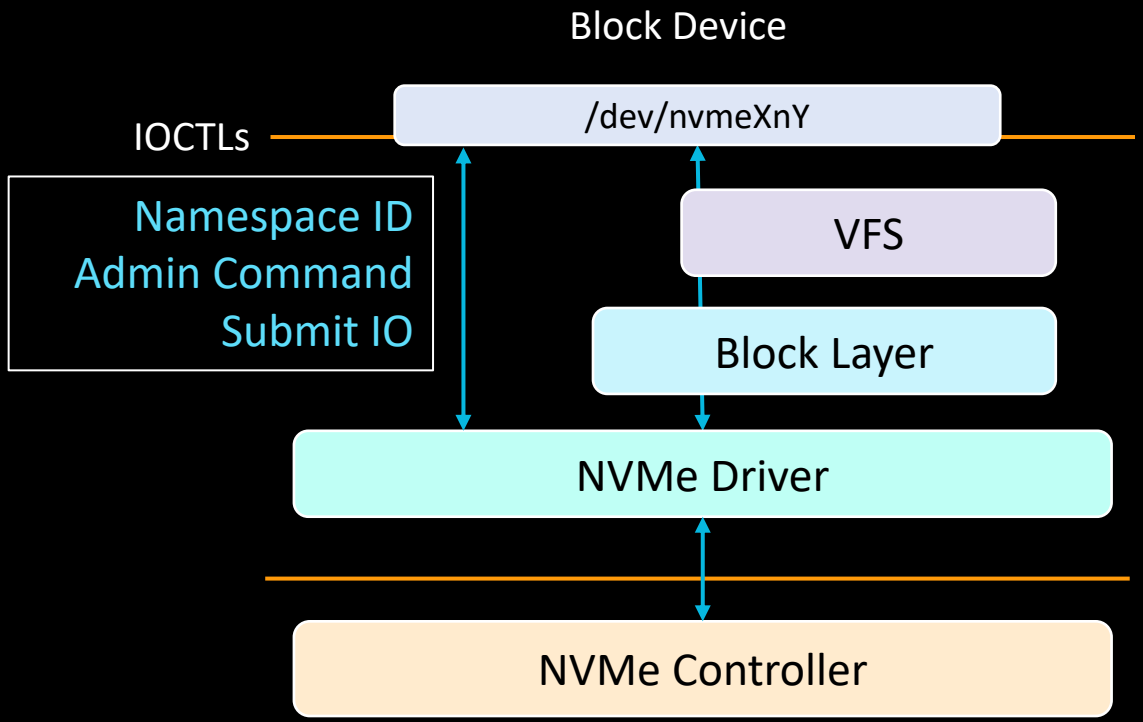
- New Namespace Types
  - Computational
  - Key-Value
  - Zone
- More logs, identifications and features
- New Commands and Command Sets
- Spinning media
- More Specifications: Refactoring effort to introduce transport specific standards



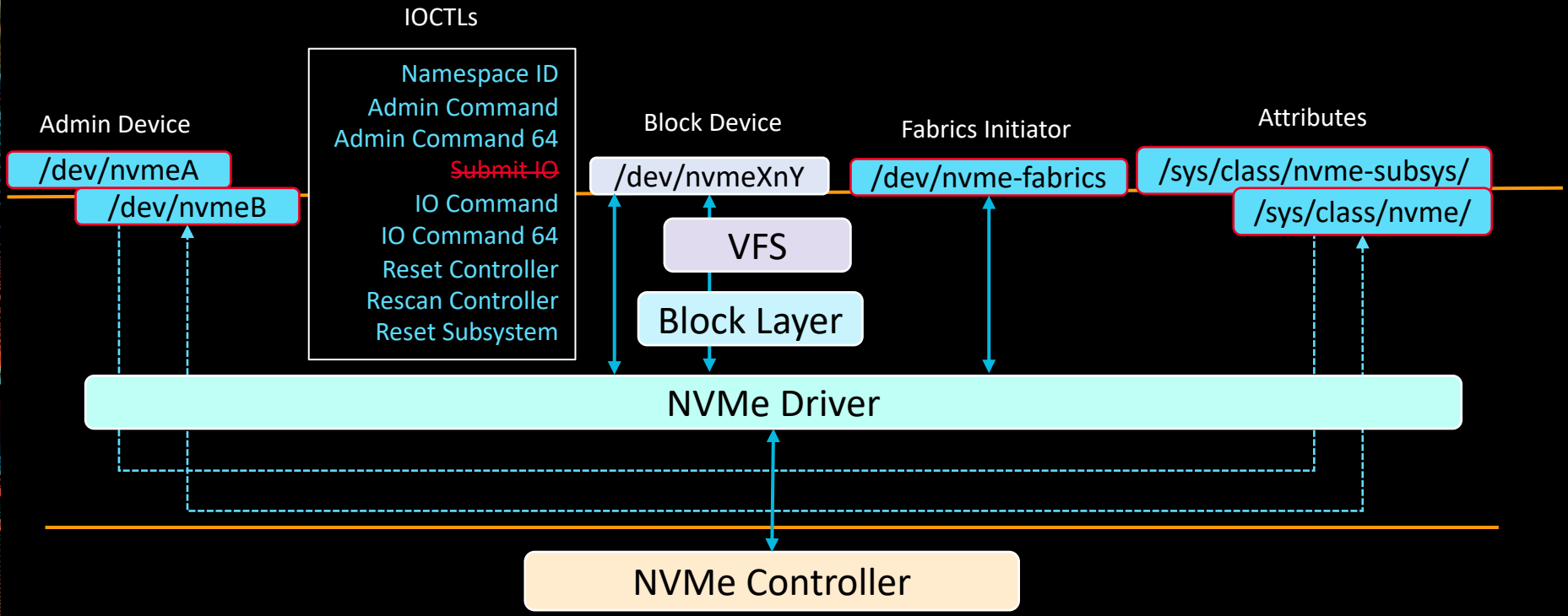




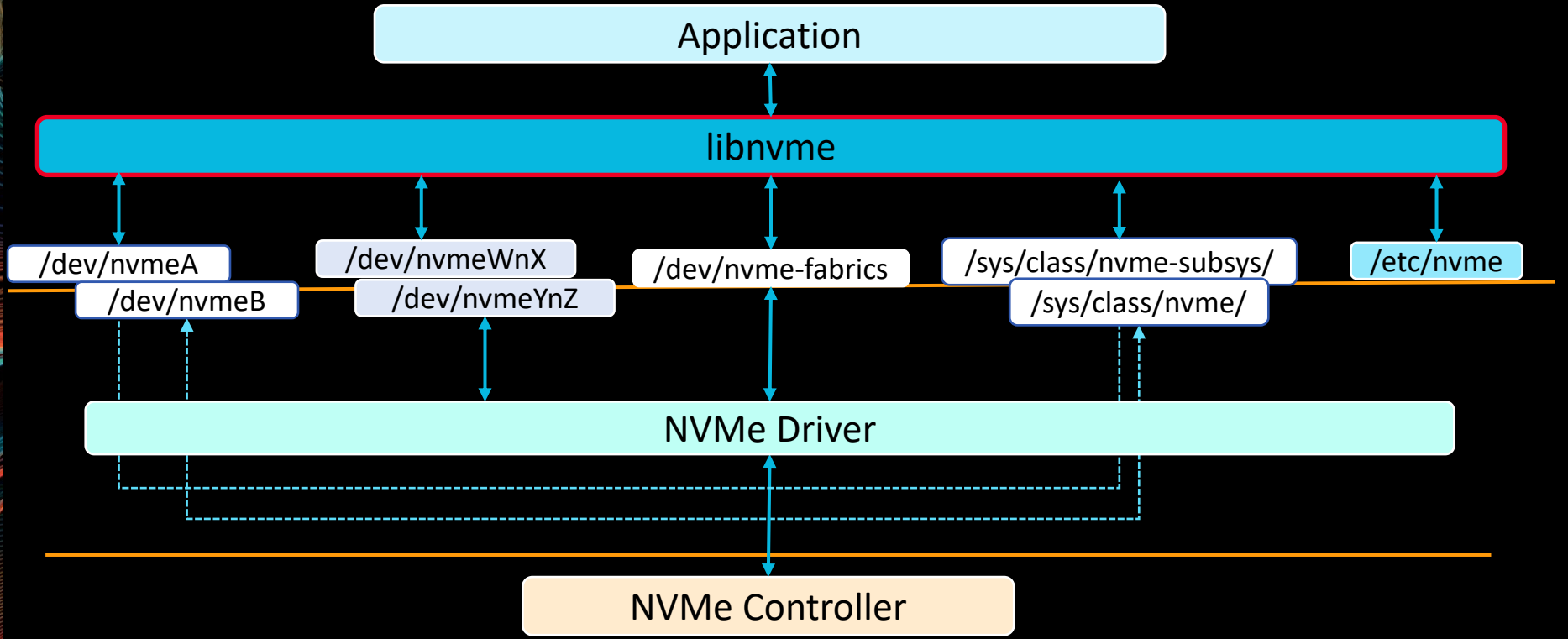
Original Linux Kernel's NVMe User Interfaces



Linux Kernel's NVMe User Interfaces Today



libnvme: One library to rule them all



Current libnvme snapshot

- Source Code Repository: https://github.com/linux-nvme/libnvme
- Language: C (extern “C” for C++)
- Code stats (approximates)
  - 20k lines of C code (mostly comments for documentation)
  - 250 exported functions
  - 170 enumerations define over 800 nvme values
  - 90 specification defined structures
- Install Artifacts:
  - Header files of API methods, structures and enumerations
  - Statically linkable archive (libnvme.a)
  - Dynamically linkable shared object (libnvme.so)
  - Documentation (html and man)

nvme/types.h

- Defines nvme data structures, fields, enumerations and bit fields
  - Currently 85 structures, and 80 enumeration types (540 unique values) defined
- Updated to match the nvme specification and ratified proposals
- Documentation describes all types and cross links to related functions, structures and enumerations

struct nvme_telemetry_log

Retrieve internal data specific to the manufacturer.

Definition

```
struct nvme_telemetry_log {
  __u8 lpi;
  __u8 rsvd1[4];
  __u8 ieee[3];
  __le16 dalb1;
  __le16 dalb2;
  __le16 dalb3;
  __u8 rsvd14[368];
  __u8 ctrlavail;
  __u8 ctrldgn;
  __u8 rsnident[128];
  __u8 data_area[];
};
```

Members

`lpi`

Log Identifier, either `NVME_LOG_LID_TELEMETRY_HOST` or `NVME_LOG_LID_TELEMETRY_CTRL`

`ieee`

IEEE OUI Identifier is the Organization Unique Identifier (OUI) for the controller vendor that is able to interpret the data.

`dalb1`

Telemetry Controller-Initiated Data Area 1 Last Block is the value of the last block in this area.

nvme/ioctl.h: Generic passthrough

- Provides ioctl definitions matching kernel UAPI
  - Except NVME_IOCTL_SUBMIT_IO: Kernel should deprecate this ioctl
- Generic method signatures provided at the lowest level of the command API
  - Provides everything needed to send any NVMe command and payload, including vendor specifics
  - Not very coder friendly

```
int nvme_admin_passthru(int fd, __u8 opcode, __u8 flags, __u16 rsvd,
                        __u32 nsid, __u32 cdw2, __u32 cdw3, __u32 cdw10, __u32 cdw11,
                        __u32 cdw12, __u32 cdw13, __u32 cdw14, __u32 cdw15,
                        __u32 data_len, void *data, __u32 metadata_len, void *metadata,
                        __u32 timeout_ms, __u32 *result);
```

nvme/ioctl.h: Command specific functions

- Parameterized for every command with fields and types specific to the commands' construction define usage
  - Easier to use compared to setting bits and bytes into unconstrained parameters
- Some commands, like logs, have many variations
  - Another layer will make it easier to reach the desired features

int **nvme_get_log**(int *fd*, enumnvme_cmd_get_log_lid *lid*, __u32 *nsid*, __u64 *lpo*, __u8 *lsp*, __u16 *lsi*, bool *rae*, __u8 *uuidx*, __u32 *len*, void * *log*)

NVMe Admin Get Log command

Parameters

`int fd`

File descriptor of nvme device

`enum nvme_cmd_get_log_lid lid`

Log page identifier, see `enum nvme_cmd_get_log_lid` for known values

`__u32 nsid`

Namespace identifier, if applicable

`__u64 lpo`

Log page offset for partial log transfers

`__u8 lsp`

Log specific field

nvme/ioctl.h: Command subtype specific

- Commands with subtypes: Get Log Page, Identify, Get/Set Features, Get/Set Directives
- Provide easy access to each sub-type of command by further parameterizing each
  - Constraining parameters is easier to use and provides type safety
  - 140 unique command functions for their specific command mode and payload
  - 85 enumerations for command parameters

int nvme_get_log_telemetry_ctrl(int *fd*, bool *rae*, __u64 *offset*, __u32 *len*, void * *log*)

Parameters

`int fd`

File descriptor of nvme device

`bool rae`

Retain asynchronous events

`__u64 offset`

Offset into the telemetry data

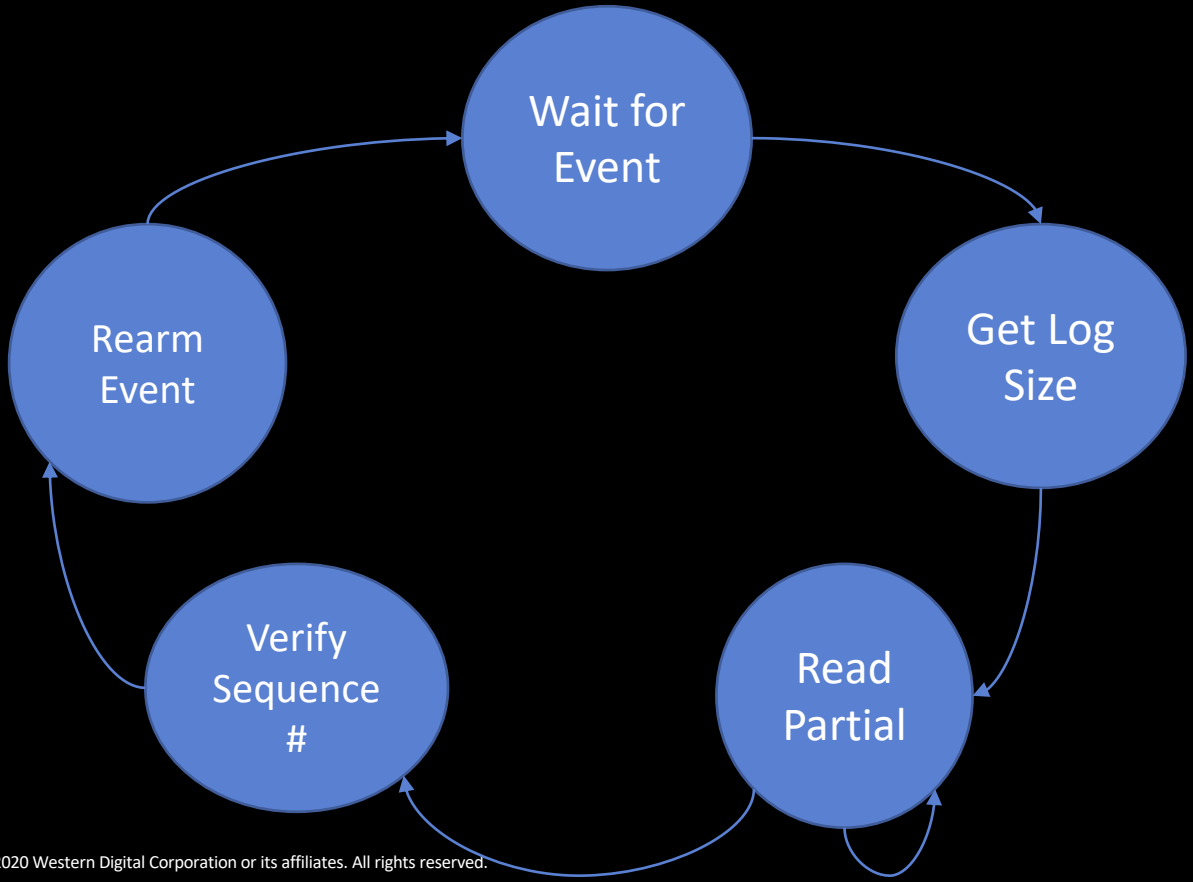
`__u32 len`

Length of provided user buffer to hold the log data in bytes

`void * log`

User address for log page data

nvme/util.h: Complex command sequences



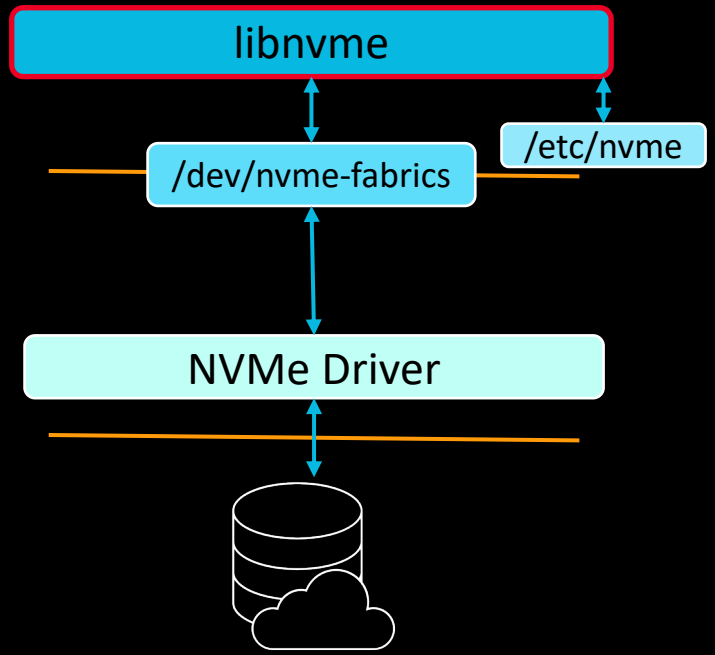
nvme/util.h: Complex command sequences

- Some information types require a longer or more complex sequence to successfully transact data: another layer to the rescue
- An incorrect sequence risks collecting a torn or stale payload
- libnvme's utils provide helper functions to manage some of complex sequences

```
int nvme_get_ctrl_telemetry(int fd, bool rae, struct nvme_telemetry_log **log);
```

nvme/fabrics.h

- Initiating connections to NVMeOF targets uses a special device handle: `/dev/nvme-fabrics`
- User API: Write magic strings to initiate
- libnvme provides helpers to:
  - Generate and submit the magic strings
  - Recursively discover and connect to targets
  - Decoding host and target configuration files in /etc/nvme/
- Updated in sync with linux-nvme driver's fabric UAPI



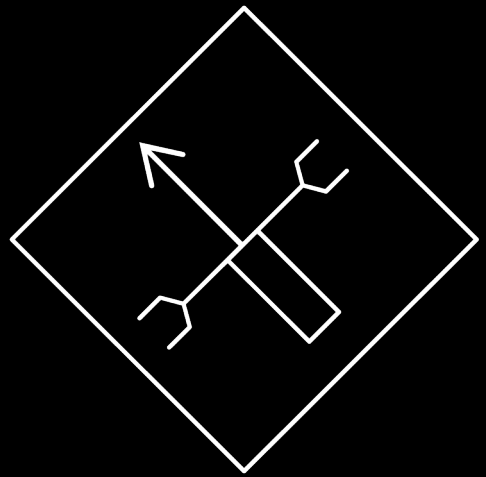
nvme/tree.h

- Establishes controller and namespace relations with each other, their parent devices, subsystem groups, and multi-path statuses
- Creates handles to all nvme objects in the system hierarchy and APIs to traverse, search, filter, retrieve attributes, and submit commands
- Filtering examples:
  - Find all controllers from vendor “ACME”
  - Find all controllers connected with RDAM transport
- Updated as needed to match the Linux driver’s sysfs hierarchy

```
|-- nvme-subsys5 - NQN=testnqn
|   |-- nvme5n1 lba size:512 lba max:488397168
|   |-- nvme7 loop  live
|   |   `-- nvme5c7n1 optimized
|   |-- nvme6 loop  live
|   |   `-- nvme5c6n1 optimized
`-- nvme-subsys0 - NQN=nqn.2018-01.com.wdc:nguid:19190E802017-0001-001B448B44AF756C
    `-- nvme0 pcie 0000:09:00.0 live
        `-- nvme0n1 lba size:512 lba max:488397168
```

Remaining work

- A few specifications features remain to be implemented
  - Persistent event log
  - Management Interface (MI) send/receive
- Uncommon features remain untested
  - Need to be verified with real hardware to confirm the library's types and commands are correctly implemented
  - Finding target support for some features can be difficult!
- New NVMe specification features are coming soon: will need to implement these
- Complete integration with nvme-cli
  - Release and request package support in our favorite Linux distributions



The End