presented by

Enabling Advanced NVMe Features Through UEFI

Spring 2018 UEFI Seminar and Plugfest
March 26-30, 2018
Presented by Zachary Bobroff(AMI)

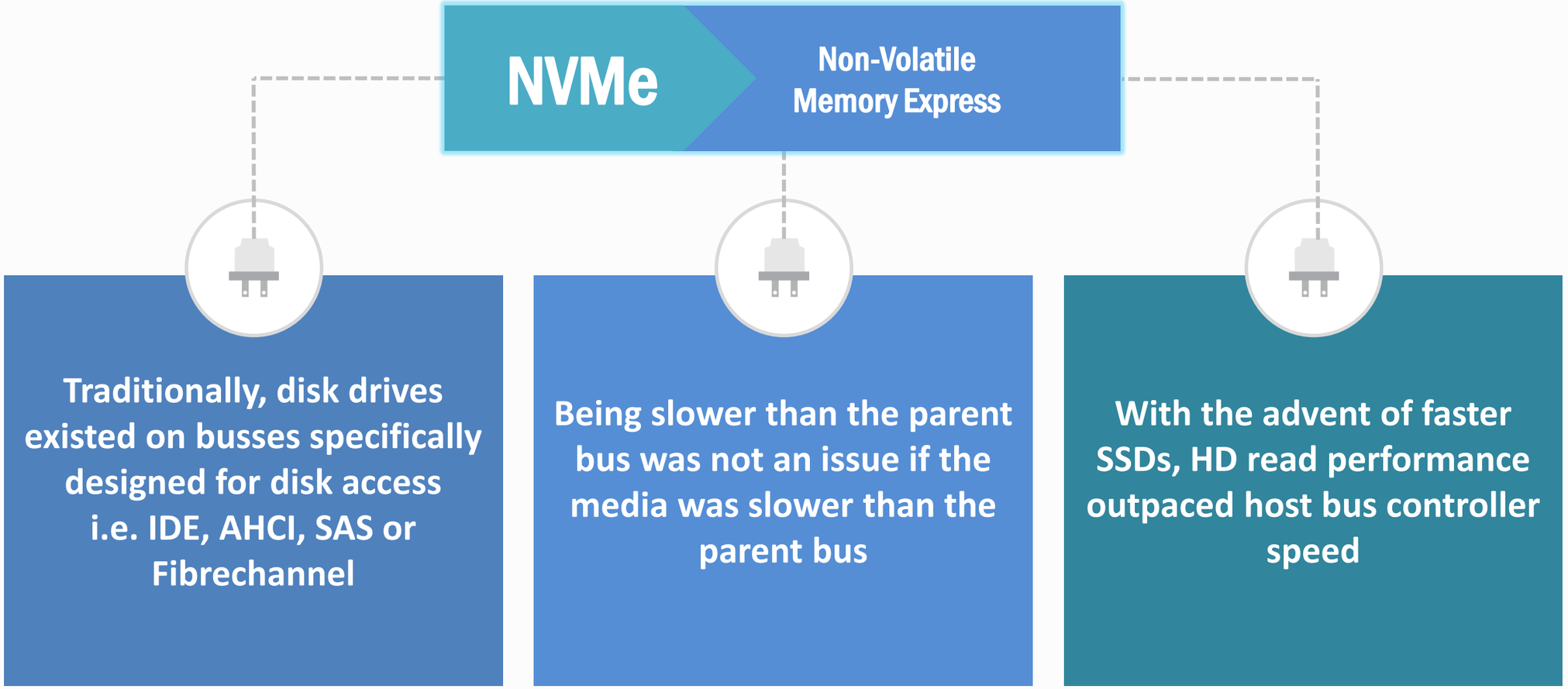
Agenda

- What is NVMe?
- What Features are Missing?
- How to Enable these Features
- Conclusions
- Questions?

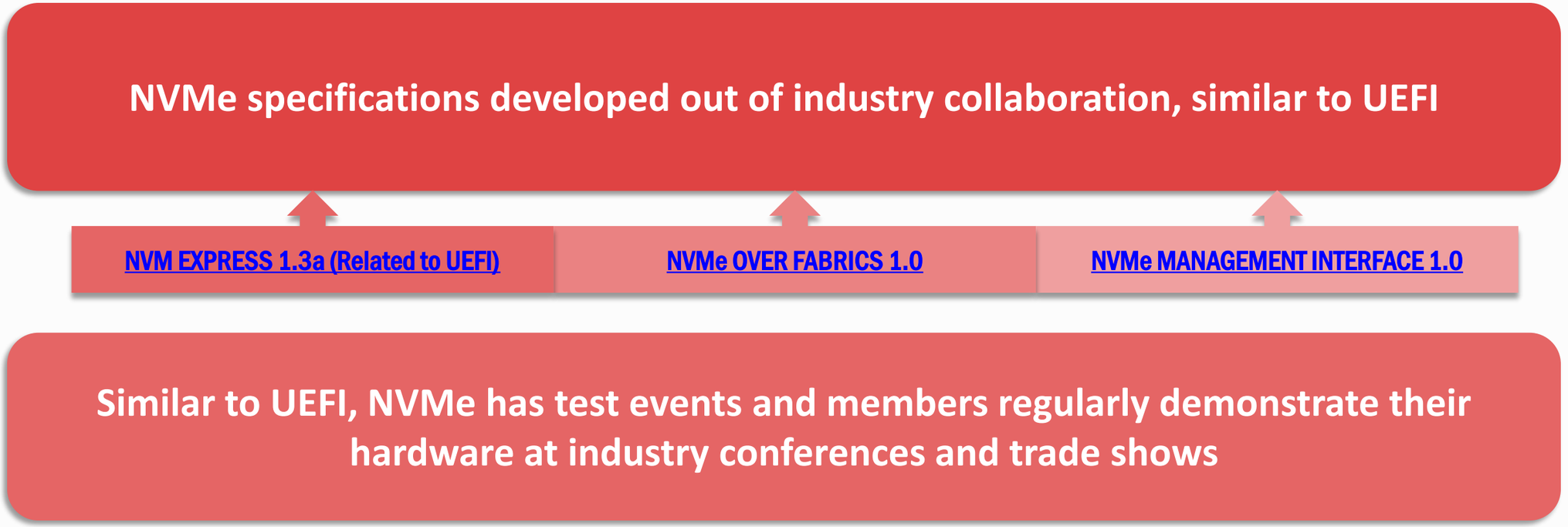
Enabling Advanced NVMe Features Through UEFI

What is NVMe?

NVMe Introduction



NVMe Specification



AHCI NVMe Comparison

	AHCI	NVMe
Latency	6.0 us	2.8 us
Max Queue Depth	Up to 1 queue with 32 commands	Up to 64K queues with 64K commands each
Multicore Support	Limited	Full
4KB Efficiency	Two serialized host DRAM fetches required	One 64B fetch

Cited from: sata-io.org

NVMe Popularity and UEFI

Increased speed capabilities, makes NVMe the most popular storage medium for high end:

- Servers
- Workstations
- Desktops
- Etc.

UEFI was also quick to adopt NVMe as a boot media

- UEFI is very standardized
  - UEFI only needs to provide basic device support through EFI_BLOCK_IO_PROTOCOL
- UEFI also provides some ancillary protocol support
  - EFI_NVME_PASS_THRU_PROTOCOL to talk directly to controllers/drives

Enabling Advanced NVMe Features Through UEFI

What Features are Missing?

NVMe Features in UEFI

- UEFI only needs to provide basic support for booting to an NVMe disk and talking directly to a controller
- Traditional hard drives had many additional features used in UEFI firmware but not commonly applied for NVMe!
- NVMe has many additional features used at OS time that firmware could also use!

Missing Traditional Hard Drive Features in UEFI

SAT3 Password Support – Ability to set a password on the drive so it cannot be used without password

Pyrite Support – Capability to self-encrypt drive so it cannot be read without proper authentication

Block SID Support – Specification that covers firmware and OS communication to handle freeze-locking events for a self-encrypting hard drive

NVMe Features Missing in UEFI

- **NVMe Namespace Creation** – Capability called “namespace” where a drive can be segmented. While generic EDKII has the ability to read namespaces, end users may want to create namespaces in pre-boot
- **NVMe Controller FW Update** – Secure way to update the host controller firmware
- **NVMe Metadata** –Metadata that exists in parallel to actual drive data; data used by drive to ensure data integrity

How Do Interested OEMs Add Support

Using UEFI interfaces, how does an OEM add support for these features on top of base features provided in their UEFI FW?

Enabling Advanced NVMe Features Through UEFI

How to Enable Additional Features

UEFI Provides the Building Blocks

UEFI Specification provides EFI_NVME_PASS_THRU protocol

All UEFI PassThru protocols provide base way to talk to a drive/controller

Using these protocols allows developers to provide support for all features described

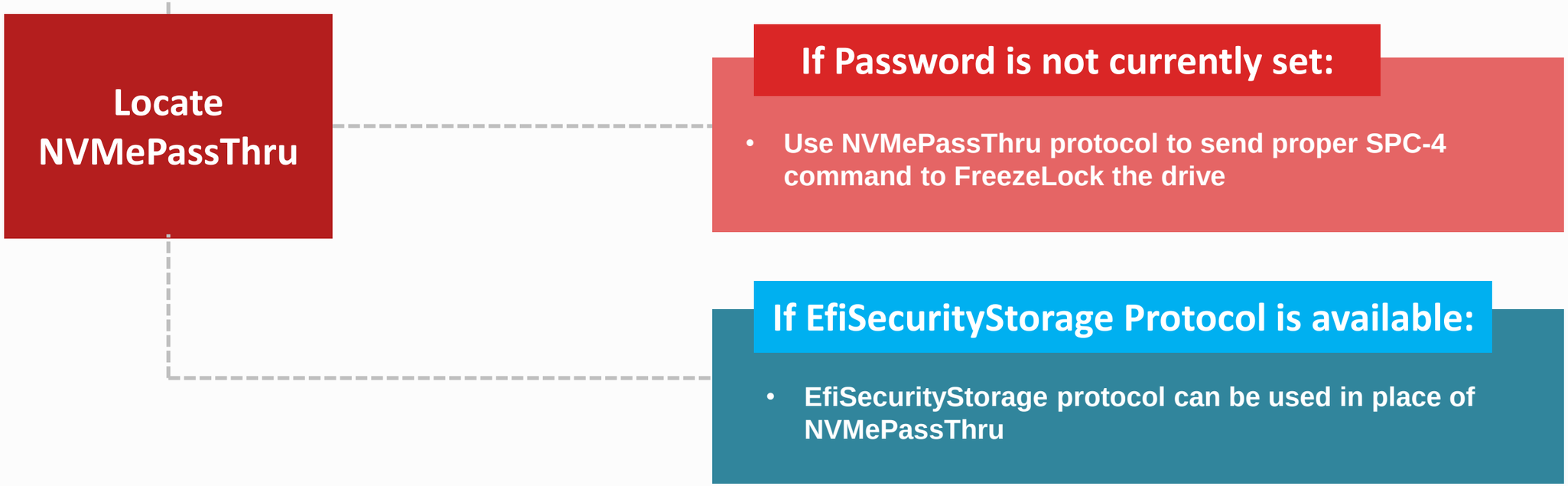
- Provided the NVMe device has support for feature being developed

SAT3 Password Support

SAT3 Password Support is a standard method of having a hard drive password; will not work if password is not supplied

- This is commonly used to marry a drive to a system
  - If removed, the data cannot be read without the password
- Even if drive has no password set, the drive should be “freeze locked”
  - Prevents malicious code from setting one later

SAT3 Password Sample Implementation



Pyrite Support

Pyrite Support is when a drive can self-encrypt specific data areas of the drive

- Protects drive data by encrypting data on disk by the drive
- Drive may have early boot loader data unencrypted and decrypt remaining area after OS authentication is complete

Pyrite Sample Implementation

Using NVMePassThru Protocol:

- Take ownership of storage device
  - Set new password to C_PIN Credential
- Activate the locking SP
- Change Admin1 Pin in locking Service Provider (SP)
  - Additional User password can also be set
- Configure locking objects (LBA Range)
  - Configure the ranges (regions) that will be locked (encrypted). Entire disk is also an option.

On subsequent boots, password must be applied to drive to unlock regions locked above

Block SID Support

Block SID Support is a TCG specification that manages “freeze locking” or not “freeze locking” of certain drive features as requested by the OS

- Spec requires validation where someone is physically present, but allows OS to request the drive remain unlocked so OS can begin drive encryption on next boot
- BlockSid support can be detected using level0 discovery command with feature code 0402
  - This will return if Block SID is supported and if it is currently locked
  - Firmware should report if any media supports Block SID

BlockSID Sample Implementation

On every reboot, check for OS mailbox event

If no mailbox event from OS:

- Use NVMePassThru protocol to FreezeLock the drive
- Refer to TCG storage specification for proper command information in table marked “Block SID Authentication Channel”

If mailbox event from OS:

- Check for user presence (some pop-up is the easiest way)
- Continue boot with no additional commands and OS can take ownership of the drive

Multiple Namespace Support

Multiple Namespace Support provides support for namespaces where one drive can appear as two different NVMe devices

- Namespaces provide a good way to better utilize NVMe storage media
- NVMe namespaces can be setup within the OS, but what if someone wants to do this before installation?

Multiple Namespaces

Provide User Interface in Setup for displaying, deleting and creating namespaces

Use NVMePassThru protocol to talk to NVMe drive through NVMe Attachment/Management commands

- Refer to NVMe specification for additional details on this command

Default NVMe driver will automatically create a BlockIO instance for each namespace discovered

Controller FW Update

Controller FW Update enables NVMe device controller firmware to be updated

- Updating of firmware should be done in a safe and secure environment
- Using NVMe PassThru and Capsule update allows a capsule to be provided to the UEFI FW
- Provides standard method to accomplish secure FW updates for all NVMe devices

FW Update Sample Implementation

On reboot, firmware checks for capsule

If no capsule exists for NVMe controller

- Continue booting normally

If capsule exists for NVMe controller:

- Verify/authenticate capsule image
- Push capsule to the drive using NVMePassThru Protocol via “Firmware Image Download” command & “Firmware Commit” command
- Reboot system

NVMe Metadata

NVMe Metadata is kept within the drive in parallel to actual device firmware

- When data is read, metadata is compared to verify data integrity
- When data is written, metadata is written to metadata storage with proper CRC information for verification

NVMe Sample Implementation

First, drive must be formatted correctly to accommodate metadata

Once drive metadata is enabled, standard EfiBlockIo will not work on its own

EfiBlockIo protocol in the end uses NVMePassThru

- Normal NVMePassThru block read/write must be extended to accommodate additional blocks relating to metadata information
- Both read and write operations must be extended
- Extension means when reading/writing specific sector, buffer needs to be extended by length of metadata
- Read operation can check metadata for integrity before return block

Enabling Advanced NVMe Features Through UEFI

Conclusions

Conclusions

NVMe is a relatively new technology in the computer ecosystem

NVMe has been quickly & broadly adopted, some useful features may have been overlooked

UEFI provides the building blocks to provide support so any developer can contribute!

What other technologies can be better extended using PassThru protocols?

Questions?

Thanks for attending the Spring 2018 UEFI Plugfest

For more information on the UEFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by