UEFI Ecosystem and NVMe Device bring-up & debug under UEFI

Uma M. Parepalli

SK hynix memory solutions

Session Two Agenda

- **Software Ecosystem (90 min) – Keith Bush/Kwok Kong/Uma Parepalli**
  - UEFI BIOS – (40 min) Uma Parepalli
    - Introduction to UEFI environment, drivers, BIOS vendor eco system and UEFI platform firmware
    - Bringing up and debugging NVMe devices from power on, OS boot to shutdown/reset
    - Validating under UEFI
  - NVMe drivers – (45 min) Keith Busch: Linux (20 min) Kwok Kong: Win/ESX (25 min))
    - Status Update
    - Driver Architecture
    - Driver Features
    - Future Work and Directions
- **Management Interface (30 min) – Austin Bolen**
  - Architectural Model
  - Sideband Management Stack Overview
  - Physical Interconnect – Connecting BMC to the NVMe device
  - Protocol Layer
- **Conformance and Interoperability (30 min) -- David Woolf**
  - NVMe Conformance Suite
    - Usage
    - Results
    - NVMe Tools
    - Best Known Methods
  - NVMe Interoperability Suite
    - Best Known Methods
    - NVMe Integrators List

Scope

- UEFI Platform Firmware Ecosystem
  - OEMs, OS & BIOS Vendors and IHVs
  - UEFI Driver
- Bringing up, Debugging & Validation
  - As an add-on secondary SSD
  - As a primary boot SSD
  - Power-on, UEFI & OS boot, Shutdown/Reset
  - Tools Available
- Q&A
- Backup – What is UEFI?

UEFI Platform Firmware & NVMe Ecosystem

UEFI & NVMe Ecosystem

- Consists of OEMs, OS, BIOS and Hardware Vendors
- NVM Express & UEFI Forum
- Validation Test Suite Providers
  - UNH, WHCK, UEFI SCT
- Tools Providers
  - PCIe NVMe Tracers
  - Source Code Analyzer HW Tools
- Customers

UEFI & NVMe Ecosystem..

- There is no separate forum for UEFI NVMe Driver at this time
- Issues addressed in USWG/UTWG as needed
- IHVs need to work with OEMs, OS & BIOS Vendors

UEFI NVMe Driver

- UEFI NVMe Driver source is provided by Intel for you to get started
- Intel UEFI Platform Firmware requires minor changes to support NVMe devices
- Requires fixes in the Platform Firmware to handle EFI_UNSUPPORTED return status
  - Assert due to NVMExpressDiskInfo.c: DiskInfo→WhichIde()

UEFI & NVMe Ecosystem..

- Loading UEFI NVMe Driver enables File System on the NVMe Disk for BIOS to locate the boot loader
- UEFI NVMe driver is required
  - for booting any latest OS from NVMe Disk
  - for OS independent validation under UEFI Shell
  - for Factory Diagnostics & POST

UEFI NVMe Driver

- UEFI driver is yet to be widely incorporated in to the UEFI platform firmware as SATA support is incorporated
- Have the BIOS Vendors and OEMs integrate the UEFI driver in to the UEFI platform firmware.
- Validate your NVMe device at UEFI Plugfest
- Work with BIOS, OS Vendors and OEMs during development

UEFI NVMe Driver..

- UEFI Drivers can be loaded in many ways
  - Built-in to UEFI Firmware by BIOS Vendors/OEMs
  - OEM UEFI Platform Firmware volume
  - From Hardware Vendor OptionROM
  - From USB Stick – for development & debug testing only
- Your driver may not be loaded, if your device is not a boot device, in order to reduce system boot time

Customizing UEFI NVMe Driver

- Start with UEFI NVMe driver
- Add your features
  - Firmware Update – UEFI Firmware Management Protocol
  - Device Configuration and User Interface – UEFI HII
  - Diagnostics & Self Tests – UEFI Driver Health Protocol
  - Avoid heavy NVMe vendor specific customization for universal compatibility

UEFI Resources

- UEFI.ORG

Unified Extensible Firmware Interface Forum

- Source Forge
  http://tianocore.sourceforge.net/wiki/Welcome

- http://nvmexpress.org/drivers/

Drivers on NVM Express site

Drivers

- Linux Driver
- Windows Driver
- UEFI Driver
- FreeBSD Driver
- VMware Driver (in development)
- Solaris Driver (in development)

Bring-up & Debug Requirements

Bring-up & Debug Setup - System Requirements

- Latest UEFI Development Vehicle - Desktop or Server with Serial output console
- Need a Serial-to-USB cable
- Primary SAS/SATA OS Bootable Disk
- MS Windows OS 8.1 & Ubuntu Linux 14.04 64-bit DVDs
- Windows 8.1 OS system
- UEFI Tools such as diskpart, efifmt
- Debug ports to connect ITP
- PCIe slots for PCIe NVMe Tracer/Analyzer

Bring-up & Debug Setup – NVMe Controller Requirements

- Debug port for firmware debug logs
- PCIe NVMe Analyzer with NVMe Trace & NVMe decode capabilities with large trace buffer
  - For creating several FAT/FAT32 and other partitions
  - for testing OS installs

Bring-up & Debugging as a secondary SSD

Bring-up & Debugging as a Primary Boot SSD

Bring-up & Debugging as a secondary SSD

- Load & Verify UEFI Driver under UEFI Shell
- Create & Format FAT32 Volumes under OS
- Perform basic File I/O under OS
- Mount those FAT32 Volumes under UEFI
- Perform File I/O under UEFI
- Copy & execute OS loader and boot OS
- Run UEFI SCT

Load & Verify UEFI Driver

- Power-on system, boot to UEFI shell
- Verify the device/controller is detected
  - Use "*pci -b*" command to look for your Vendor ID & Device IDs
- Download and Build UEFI Driver
  - Flash *driver.rom* onto OptionROM (or)
  - Copy *driver.efi* to USB Stick
- "*load driver.efi*" from USB stick if no OptionROM

Load & Verify UEFI Driver ..

- Verify that driver is loaded successfully with "*drivers*" command
- Use "map –r" to mount the raw disk
- Use "dh", "*devices*" and "*devtree*" commands to check further
- Use UEFI "*diskpart"* to GPT Initialize
- "*exit*" and boot to OS from other boot disk

Load & Verify OS Driver ..

- Verify that OS driver is loaded successfully and device is recognized.
- Check the Disks and Storage Controllers under Device Manager
- Test disable/enable, uninstall/discover functions
- Initialize NVMe disk to GPT using Disk Management

Create FAT32 Volumes

- Create and format FAT32 volumes under OS
- Perform File I/O under OS
- *Shutdown* OS and boot to UEFI Shell
- *"load"* UEFI driver and verify driver load
- Mount these FAT32 volumes under UEFI shell with "*map –r"* command
- You should find your NVMe device as a file system, for example "fs1".

Boot to OS as a secondary SSD

- Verify the volume contents – “dir fs1:”
- Perform File IO operations under UEFI Shell
  - Use “*dir*”, “*copy*”, “*delete*” commands
- Copy OS loader from other boot disk with “*copy –r*”
- Execute OS loader and let it boot to OS
- Shutdown/Restart and boot to UEFI Shell
- Verify FAT32 Volume Contents

Bring-up & Debugging
as a secondary SSD

Bring-up & Debugging
as a Primary Boot SSD

OS Installation & Boot

- Install Windows 8.1 or Ubuntu Linux 14.04 64-bit DVD on to the NVMe disk
- Watch the OS install specific reboot operations, may require UEFI Driver
- Boot to UEFI Shell and execute OS loader

"*Efi\boot\<bootloader>.efi*"

- Watch the OS boot process related I/O
- Create & Format another FAT32 volume
- Perform File I/O under OS & UEFI Shell

Debugging Tips

Debugging Tips

- Enable debug in the UEFI environment
- Enable serial console and log the console output
- Enable debug in your NVMe device through UART or some other means for NAND/Host FW logs
- Analyze UEFI & FW logs and PCIe NVMe Trace
- Enable debug at OS level too, if possible

Debugging tips under UEFI Shell

- Previously created FAT/FAT32 partition may not get mounted correctly under UEFI shell.
- Verify that there is no disk corruption and/or data loss between reboots
- Use *load/unload, drivers, dh, devices, devtree, map* and *dblk* commands
- Use *dblk* to dump the LBAs to check for the existence of partition structure
- You cannot mount NTFS partitions under UEFI

Debugging Tips – Contd..

- Crash during UEFI driver load
  - Could be UEFI Platform firmware issue
  - Debug through UEFI Driver
- Crash under MS Windows
  - Seek and delete phantom disks & devices under MS Windows device manager
    - Disk Drives
    - Storage Controllers
- Analyze the logs & traces

Standards Compliance under UEFI & Operating Systems

UEFI SCT

Windows HCK

UNH, PCI SIG

UEFI Software Compliance Test (SCT) Suite

- UEFI SCT can be downloaded free from Source Forge
- Supports OEM Device Path Test only
- Read instructions to start tests
- Create a FAT32 partition on another disk and use it under UEFI to run the UEFI SCT
- Will take longer if running from USB Stick
- Examine logs, contact UTWG for help

Windows Hardware Compliance Kit (WHCK)

- Download latest WHCK and update Filters
- Follow instructions to setup Test Server/Manager and Client/Target with NVMe
- Poor and incorrect documentation
- Have two name spaces, one formatted to NTFS and one raw
- Requires frequent manual testing
- Focus on sleep and reboot tests

Windows Hardware Compliance Kit (WHCK)

- Ordering the tests
  - Basic and NVMe tests
  - Reliability, Functional and Certification tests
  - Sleep related tests
  - 4 to 8 Hr. tests
- Some 1 or 2 minute tests may take much longer

Questions?

Uma M. Parepalli

umaparepalli@gmail.com

Backup

What is UEFI?

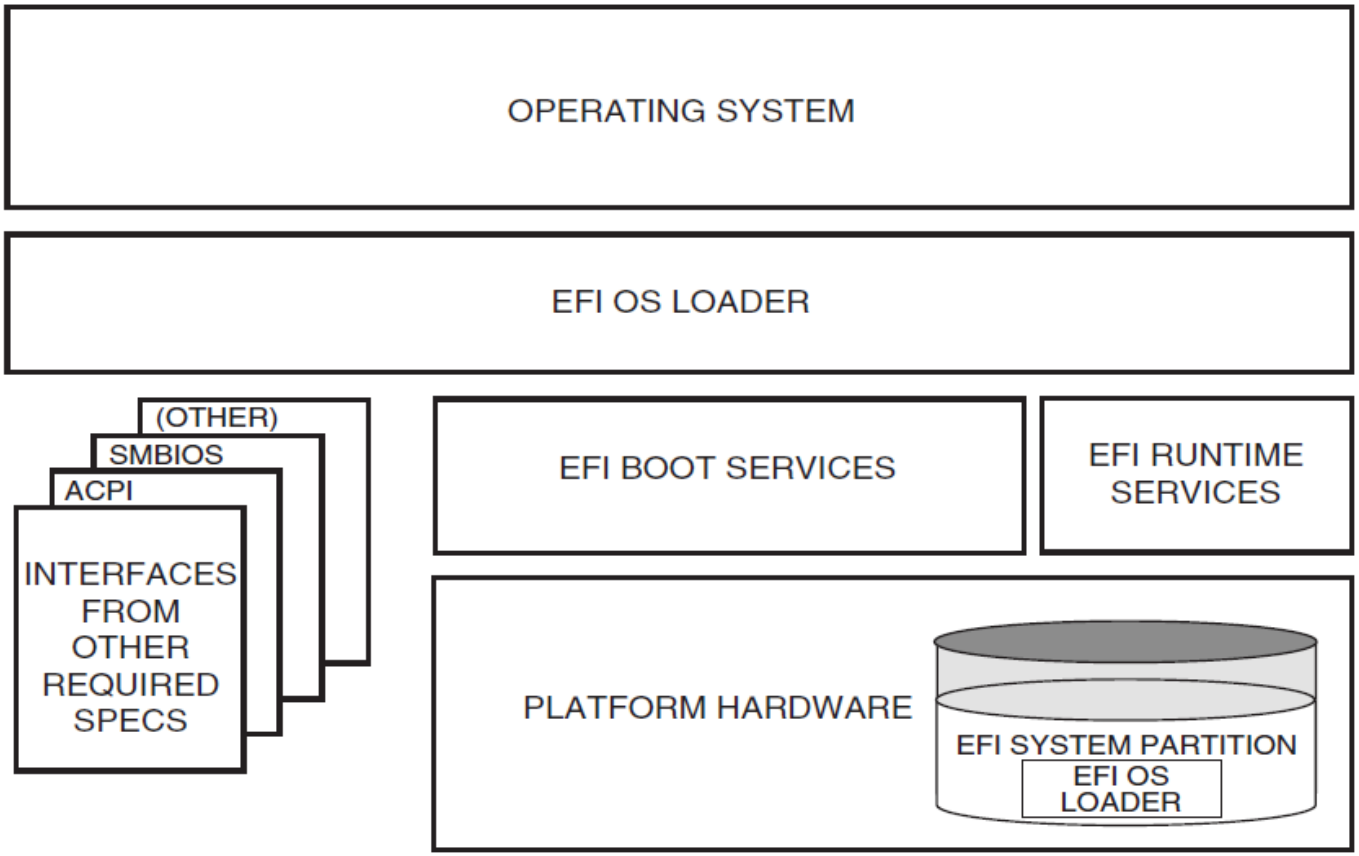
What is UEFI? http://uefi.org

- Replaced legacy BIOS
- New interface model between Platform Firmware and Operating Systems
- Provides Boot & Run-time Services for Applications, Drivers, OS boot loaders and Operating Systems
- Covers entire boot process from Power-on/Reset, OS boot to OS Shutdown

What is UEFI? http://uefi.org

- Loads & runs pre-boot applications & drivers
  - Factory Diagnostics, POST
- Configuration, Diagnostics, Firmware Update and Boot Device Selection (BDS)
- Has rich OS like features for FW dev. & test
- Support for processor independent EFI Byte Code (EBC) drivers
- ACPI is under UEFI Forum now

UEFI Conceptual View



UEFI Firmware Phases

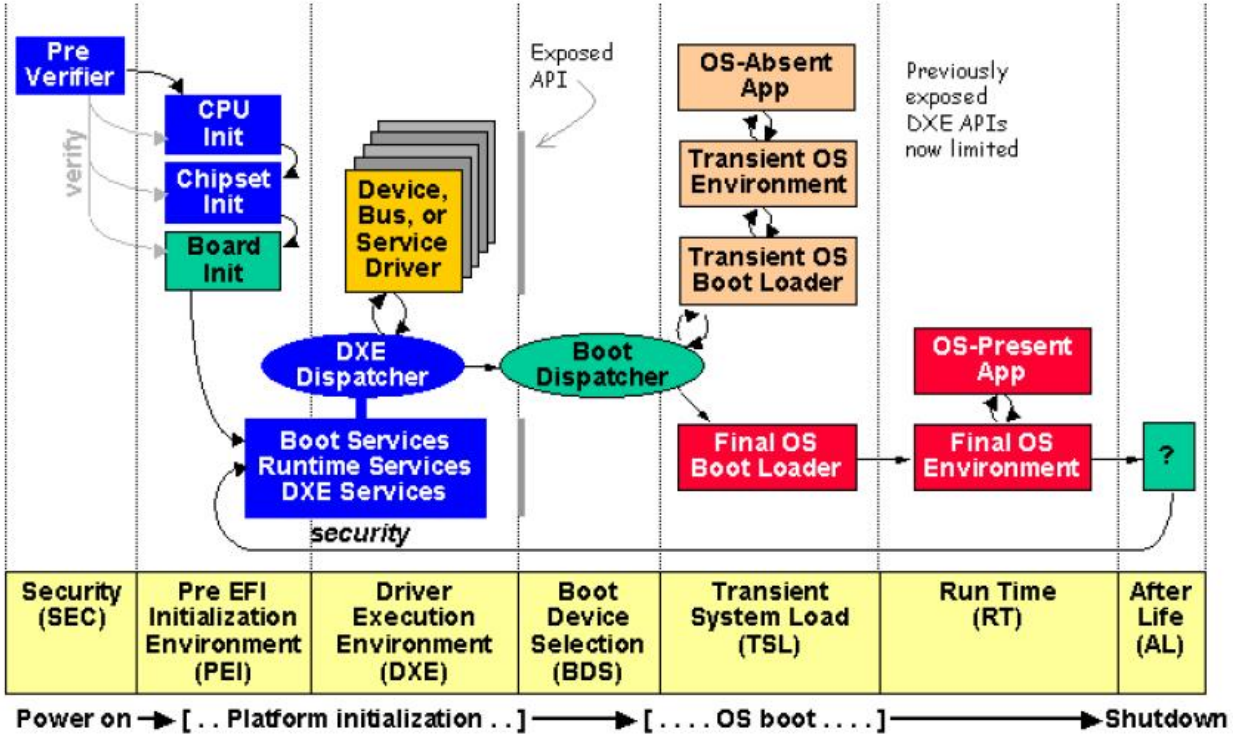
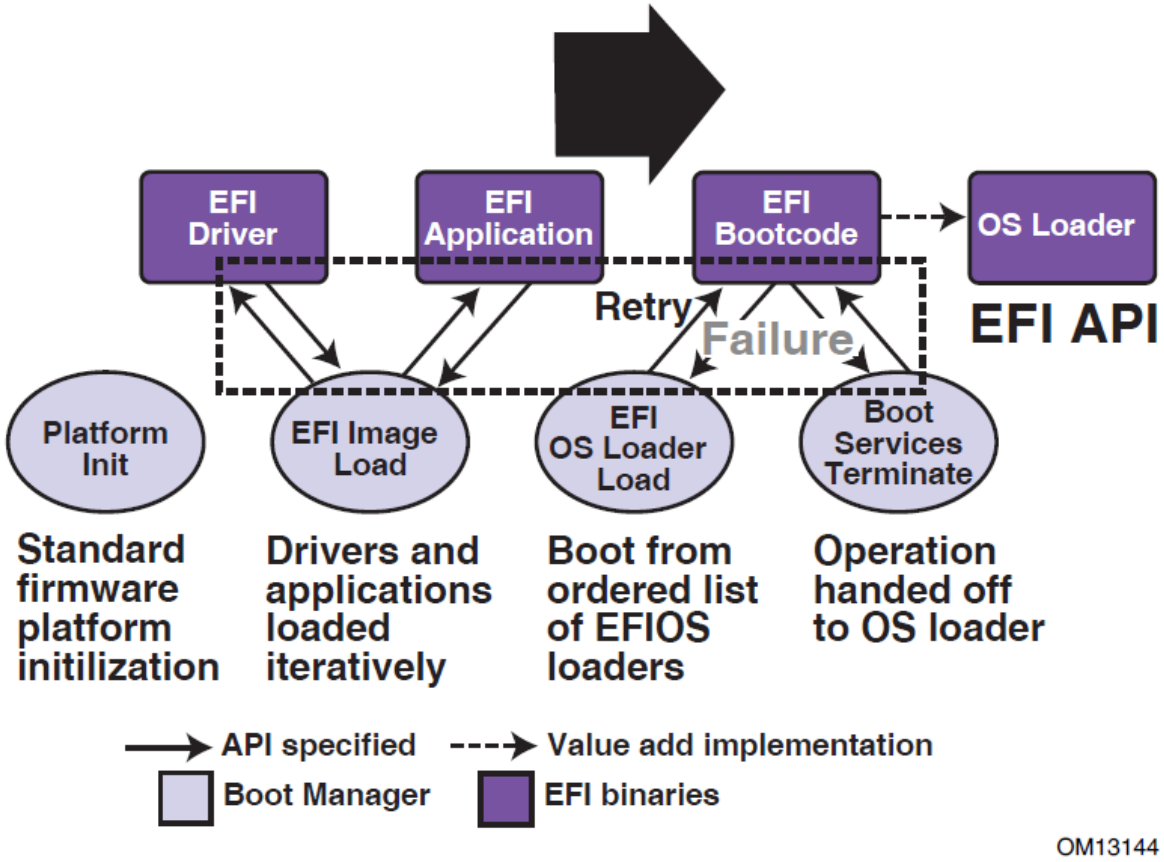


Figure 1. PI Architecture Firmware Phases

Booting Sequence



UEFI Spec & NVM Express

- UEFI Spec Rev 2.4.B has NVMe Device Path information which is covered by UEFI SCT
- Protocols – Device Path Protocol
  - Device Path Nodes
    - Messaging Device Path
      - NVM Express namespace messaging device path node
- Device Node Table
  - Type: 3 (Messaging Device Path) SubType: 23 (NVM Express Namespace)