LinuxBoot Integration With UEFI

UEFI 2020 Virtual Plugfest
June 17, 2020
Presented by Jonathan Zhang, Facebook & Isaac Oram, Intel

www.uefi.org
Meet the Presenters

Isaac Oram
Principal Engineer
Member Company: Intel

Jonathan Zhang
Software Engineer
Member Company: Facebook
Agenda

- LinuxBoot Introduction
- Proposal
- Call for Action
Why Use Linux to Boot

• Productivity: turn Linux engineer into Firmware engineer
• Manageability: black box → white box
• Update Turn Around Time: Firmware is part of platform ownership
• Security: One set of driver to harden between FW and OS
LinuxBoot Model

SPI Flash

UEFI PEI

coreboot romstage

u-boot SPL

Firmware HW init

Memory initialized

LinuxBoot

Linux Kernel

initramfs

Operating System
What is LinuxBoot?

• Initial bootloader (UEFI, coreboot):
  – Silicon, platform, and board init
  – PCIe enumeration and resource assignment
  – ACPI/SMBIOS

• Linux (kernel/initramfs) in FLASH:
  – Device Drivers (device/networking/file system)
  – Shell
  – Target OS bootloader

• u-root (a type of initramfs):
  – Busybox like
  – Go-lang scripting environment
Why is LinuxBoot Gaining Traction?

• Why now?
• Flash size is less of a problem now
• Linux becomes mature for embedded environment
• Reduce FW/OS duplication
• Hyperscaler scalability challenges
Open System Firmware (OSF)

- Open Compute Project (OCP) incubated OSF in 2018; Formalized in 2019
  - Opencompute.org

- Problem Statements:
  - OCP platform hardware design is open
  - OCP platform adoption increasingly blocked on ability to obtain/customize firmware
  - Hyperscalaler scalability challenges increase

- Activities:
  - OCP OSF requirement for OCP platform submission
  - OCP OSF checklist

www.uefi.org
OCP Open System Firmware (OSF) on Open Compute Platform

Transition schedule for Open System Firmware (OSF) on Open Compute Platform

A proposal from the OSF workstream to the OCP IC, July 26, 2019

Key Objective:
The key objective is to define clear requirements and a timeline for a logo program that ensures OCP has open firmware ready solutions widely available by 2021. The requirements must be clear and testable. The timeline must be reasonable and achievable for hardware vendors and the open communities for OCP, coreboot, LinuxBoot, and TianoCore. The logo program must support the OCP charter and goals.

Background and Introduction:
Open System Firmware (OSF) is a work stream in OCP. While there are many components on OCP systems containing firmware, OSF is directed at the primary CPUs on the board, e.g., the x86 CPUs on x86-based servers and networking equipment. While there may be support CPUs on these boards (e.g., BMC or network card) they are not currently in the scope of OSF.

OCP OSF checklist (project DRAFT)

Summary
This document is in draft status and is being reviewed within the OCP OSF (Open System Firmware) project.
This document captures requirement for the OCP OSF checklist for March 2021 OCP platform submission related to OSF, according to https://docs.google.com/document/d/1FAF81ec4K41UcDA5sUj483M8ElvDx8kAsp8BF3UCb7/w/edit#heading=h.ybzm7zc2jy69.
This checklist is intended to enable OCP adopters to use OCP platform with a basically working open system host firmware. This checklist is the initial one; more items/coverages will be added to future checklists, as the industry becomes more mature in terms of OSF.
Explanation/Execution of this checklist is responsible by OCP OSF project leads or their representatives.

What it covers
- It contains minimum items to meet above stated goal.
- It applies to all firmware design approaches, such as open EDKII, or coreboot/Linuxboot, freeboot/jetboot, to name a few.
- It applies to all architectures, including x86, ARM, POWER, etc.
- It applies to compute/storage servers and networking servers.
- It applies to host firmware.
Intel/Facebook Joint POC

• POC on Intel® Xeon® Scalable processor
  – Product formerly codenamed Skylake
• POC on first OCP multi-socket server platform
  – TiogaPass (80 threads, 384GB DRAM)
  – Stepping stone for later generations
• Coreboot support for Xeon-SP was enabled first time
  – Code upstreamed
• Coreboot utilizes Intel® Firmware Support Package (FSP)
  – Intel FSP 2.1 API mode
Intel® FSP 2.1 Specification

• Intel® FSP encapsulates basic silicon initialization for use by any bootloader
  — Contents are Intel edk2 silicon initialization PEIM

• Supports two modes
  — API mode for non UEFI bootloaders like coreboot
  — Dispatch mode for UEFI bootloaders like edk2
  — One Intel FSP binary supports both modes

• API mode is simpler
• Dispatch mode is more capable
• Intel planning MinPlatform reference board open implementation that supports both modes
• FB plans to enable coreboot as FSP API mode bootloader

www.uefi.org
How Target OS is Booted

• How LinuxBoot boots Linux?
  – Kexec
  – CSM mode
  – Google POC
• How could LinuxBoot boot Microsoft Windows?
  – Google POC
  – A small set of requirements are imposed by Windows
• What about ACPI and SMBIOS?
• What about UEFI variables?
• What about other firmware run time services?
• What about SMM?
LinuxBoot progress: boot anything from Linux — Chris Koch, Google — Platform Security Summit 2...

www.linuxboot.org

LinuxBoot progress: boot anything from Linux

Chris Koch (@hugelgupf)
Ofir Weisse

Platform Security Summit - October 1, 2019

with

Ron Minnich, Ryan O’Leary, Gan Shun Lim, Max Shegal, Trammell Hudson, Jean-Marie Verdun, David Hendricks, Andrea Barberio, Philipp Deppenwiese and many others

@hugelgupf

www.uefi.org
Google POC Findings

- Discovered minimum requirements imposed by the Windows boot manager
- 8 services
- 6 protocols

Table 13. Required UEFI Implementation Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFI_SYSTEM_TABLE</td>
<td>Provides access to UEFI Boot Services, UEFI Runtime Services, consoles, firmware vendor information, and the system configuration tables.</td>
</tr>
<tr>
<td>EFI_BOOT_SERVICES</td>
<td>All functions defined as boot services.</td>
</tr>
<tr>
<td>EFI_RUNTIME_SERVICES</td>
<td>All functions defined as runtime services.</td>
</tr>
<tr>
<td>EFI_LOADED_IMAGE_PROTOCOL</td>
<td>Provides information on the image.</td>
</tr>
<tr>
<td>EFI_LOADED_IMAGE_DEVICE_PATH_PROTOCOL</td>
<td>Specifies the device path that was used when a PE/COFF image was loaded through the EFI Boot Service LoadImage().</td>
</tr>
<tr>
<td>EFI_DEVICE_PATH_PROTOCOL</td>
<td>Provides the location of the device.</td>
</tr>
</tbody>
</table>
Bootloader and Payload Flexibility

- More choices emerging
- UEFI plays a vital role
- Linux will play a vital role in the future
Proposal

- LinuxBoot and UEFI: two communities working together, could and should
- ACPI/PI: serve both
- UEFI: reduce minimum requirements (UEFI spec 2.8 section 2.6)
- Linux: Upstream support for such minimum requirements
- End result: LinuxBoot becomes UEFI system
UEFI Proposal

- Linux OS boot is very flexible
- Linux UEFI needs can be smaller than UEFI 2.6 minimum requirements
- Proposal: reduce minimum requirements (UEFI spec 2.8 section 2.6)
  - Considering an embedded OS loader target
  - Reduce driver model requirements
    - No console or boot devices required
  - No User Interface, localization, setup application
Call for Action

• Compatible solutions with same standard vs. competing solutions
• Work together with LinuxBoot community to enable new technology
• View open source as an opportunity
Questions?
Thanks for attending the UEFI 2020 Virtual Plugfest

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

presented by

www.uefi.org