More Performant Cluster State Management
Using Open Source Firmware and a Kraken

Devon T. Bautista, J. Lowell Wofford
Los Alamos National Laboratory
01 June 2021
A Little Background on Firmware
### BIOS mechanism:

- **Mundane**
  - Control given to first boot loader found
  - Blindly executes

- **Proprietary**
  - Sans Libreboot, Coreboot

- **Limited**
  - 16-bit real-mode addressing
  - Operates in up to 1 MB of space
  - Programmed in assembly language
  - Can only address up to 2.2 TB drives

- 2005: “Unified Extensible Firmware Interface”
Towards Open Source Firmware
The 2½ “Hidden OSes” on x86

<table>
<thead>
<tr>
<th>Code you know about</th>
<th>Code you don’t know about</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring 3 (User)</td>
<td>Ring -2 kernel and ½ kernel Control all CPU resources. Invisible to Ring -1, 0, 3</td>
</tr>
<tr>
<td>Ring 0 (Linux)</td>
<td>SMM ½ kernel. Traps to 8086 16-bit mode.</td>
</tr>
<tr>
<td>Ring -1 (Xen etc.)</td>
<td>UEFI kernel running in 64-bit paged mode.</td>
</tr>
</tbody>
</table>

X86 CPU you know about | X86 CPU(s) you don’t know about
Redundant Drivers

EDK2 Firmware

Filesystem

USB

Network

GRUB Bootloader

Linux Kernel

// SPDX-License-Identifier: GPL-2.0
* #ifdef _FAT_H
* #define _FAT_H
* #include <linux/fat.h>
* #include <linux/nls.h>

/* drivers/usb/core/usb.c */
* (C) Copyright Linus Torvalds 1999
* (C) Copyright Johannes Erdfelt 1999-2001
* (C) Copyright Konrad Stolze 2001

/* net/ipv6/tcp.c */
* (C) Copyright Linus Torvalds 2001
* INET An implementation of the TCP
* Interface as the means of cc
* Implementation of the TCP
Problems

• Redundant drivers with little-to-no code sharing
  ‣ Increased attack surface
  ‣ Loading the same drivers multiple times is slow
• The code with the highest privilege and control over hardware is audited the least
  ‣ Less frequent update/deployment lifecycle than most software and operating systems
  ‣ Proprietary, closed-source
• Reliance on vendor for updates and fixes
  ‣ A bottleneck to your production timeline
  ‣ Outsourcing development to another middleman in mitigating firmware updates/issues
Scary! What is being done?
“Let Linux do it.”
Replacing redundant, closed drivers with vetted, open ones

https://www.linuxboot.org
Open Firmware Using Linux

• One implementation of drivers

• Linux: Vetted for more than 20 years in military, consumer, and supercomputing systems
  ▸ Already running on mission-critical devices around the world
  ▸ Replace lightly-tested, closed drivers with hardened, heavily-tested, open source ones

• Bootstrap customization and fine-tuning for site-specific needs
  ▸ More on the relevance to HPC upcoming

• More people understand Linux
  ▸ Leverage existing talent/experience

• More mature tooling
Open Firmware at Facebook

• **2011**: Open Compute Project announced
• **2014**: OpenBMC: Open source baseboard management controller firmware
  ‣ Now a Linux Foundation project
• **LISA18[^4]/OSFC 2018[^5]**: Facebook uses Linuxboot in the cloud
  ‣ “Booting is hard”
    ‣ Many different types of devices now vs. one *de facto* standard then
  ‣ “More demands for firmware security”
    ‣ Measured bootstrapping
  ‣ “Provisioning is hard”
    ‣ Firmware is now more complex
    ‣ Need a robust provisioning solution
Intel
Overview

Arm is an active contributor to the EDKII project hosted by the Tianocore community.

The EDKII project is an open source project that provides a modern, feature-rich, cross-platform firmware development environment for the UEFI and PI specifications developed and maintained by the UEFI Forum.

Arm contributions make sure the EDKII project constantly keeps an up to date implementation of a UEFI compliant firmware on Arm systems.

Arm contributes to both the EDKII main repository, maintaining some core packages like DynamicTablesPkg and StandaloneMMPkg, and the EDKII platforms repository, hosting support for various Arm reference platforms as well as other 3rd party Arm-based platforms maintained by either Linaro or partners.
Open Source Firmware in HPC
Motivation

• The need to boot systems most efficiently
  ▶ Custom system initialization: greater control, finer performance
  ▶ Boot times matter due to nodes more frequently rebooting
  ▶ Vendor firmware is generic

• The need to use modern, secure protocols
  ▶ TFTP, DHCP implementations can be buggy
  ▶ HTTPS
    ▶ TLS client certificates for cryptographic root-of-trust between nodes and parent

• The desire to run an extremely minimal operating system on compute nodes
  ▶ Containerize user jobs directly on top of hardware

• The desire to potentially run a cluster state manager at a very low level to have better control of the nodes
Open Firmware at LANL

Linux kernel

+ u-root
initramfs

LinuxBoot

kraken

krakenboot

* http://kraken-hpc.io/
* https://github.com/kraken-hpc/
Sources and Further Reading

[1] Hudson, Trammell. “Bringing Linux back to the server BIOS with LinuxBoot”. 
   https://trmm.net/LinuxBoot_34c3/. 2017-12-29.
   https://www.youtube.com/watch?v=eKVSBESoKUc.
Questions?

Devon Bautista
dbautista@newmexicoconsortium.org

Lowell Wofford
lowell@lanl.gov

Join the Open Source Firmware Slack:
https://slack.osfw.dev/