Porting FreeBSD to Firecracker

Colin Percival
Tarsnap Backup Inc.
cperciva@tarsnap.com

May 19, 2023

Who am I?

- FreeBSD committer since 2004.
- FreeBSD/EC2 maintainer since 2010.
- Amazon Web Services Hero since 2019.
- I do not work for Amazon.
- I do not speak for Amazon.
- I have signed many Amazon NDAs so there are things I can't talk about.

What is Firecracker?

- Firecracker is a virtual machine monitor which uses the Linux Kernel Virtual Machine (KVM).
- Like qemu or bhyve(8) — only the userland portion of the hypervisor, none of the kernel bits.
- Developed by AWS for the AWS Lambda "serverless" compute service.
- Focus on minimalism and security.
- Apache 2.0 licensed (some portions BSD 3 clause).
- Written in Rust!

Why FreeBSD on Firecracker?

- Porting to different platforms can help to expose bugs.
- Booting in a minimalist environment can help to reveal performance issues.
- Maybe some day Amazon will support FreeBSD in Lambda?
- Because it's there!

Boot entry point

- Firecracker was designed to boot Linux.
- FreeBSD is not Linux!
- Fortunately there were Firecracker patches available to support PVH boot mode in addition to "linux boot".
- FreeBSD had support for PVH booting under Xen.
- Firecracker couldn't find the PVH entry point.
- Firecracker was looking for a `PT_NOTE` ELF note, and FreeBSD had a `SHT_NOTE` ELF note.
- A minor change to the FreeBSD kernel linker script got the FreeBSD kernel launched.

Kernel debugging without a serial console

- We don't have a serial console yet!
- No kernel panics, no printfs...
- We can see if the Firecracker process is still running and how much CPU time it is using.
- When launching a FreeBSD kernel I would get a single trit of feedback:
  1. CPU hit a triple-fault and VM exited.
  2. CPU hit a `hlt` instruction and is using 0% of CPU time.
  3. CPU is still running and using 100% of CPU time.
- Binary search through the kernel startup code to see where it's faulting!
- I realized later that `outb(ch, 0x3F8)` worked...

Xen hypercalls

- FreeBSD's PVH entry point was designed for booting Xen/PVH.
- Firecracker is not Xen and does not implement Xen hypercalls!
- Temporary workaround during development: Comment out all of the Xen hypercalls.
- Long-term fix: Check CPUID for Xen hypervisor signature and only attempt to make Xen hypercalls if we're running under Xen.

PVH memory layout

- In PVH booting, a "start info" page is provided with system metadata used during the boot process.
- Xen places this page above the kernel, while Firecracker uses a fixed low address (`0x6000`).
- FreeBSD needs some free memory very early in the kernel boot and was using "immediately after the start info page" for this purpose.
  - ... which turns out to be the page Firecracker assigned for use as the stack.
- Fix: Look at all the addresses provided by the PVH loader and set `physfree` to be after all of those structures.

Memory map

- Before FreeBSD can set up virtual memory, it needs to know how much "physical" memory it has.
- Normally the boot loader obtains a memory map from BIOS or EFI and passes it as a preloaded "module" to the kernel.
- In PVH booting, the kernel is loaded directly, without the boot loader.
- FreeBSD's Xen/PVH code obtains a memory map via a Xen hypercall, but Firecracker does not implement Xen hypercalls.
- Firecracker provides a newer version of the PVH "start info" page, which includes the memory map.
- Fix: If `start_info->version >= 1` then parse the included memory map instead of making a Xen hypercall.

No ACPI!

- On the `amd64` architecture, FreeBSD makes use of ACPI to discover hardware — including CPUs and the per-CPU "local APIC".
- Firecracker doesn't provide ACPI, instead exposing information about CPUs via the historic Intel MultiProcessor Specification `MPTable`.
- Support for `MPTable` is available via `device mptable`.
- Fix: Custom `FIRECRACKER` kernel configuration.

MPTable breakage

- Linux looks for the `MPTable` structure in the wrong place in RAM.
- Linux doesn't parse the `MPTable` structure correctly.
- Firecracker was designed to boot Linux.
- Fix: Add `options MPTABLE_LINUX_BUG_COMPAT` to FreeBSD which implements bug-for-bug compatibility.

Serial console output

- Kernel output gets printed to the serial console, but output from userland stops after 16 characters.
- I've seen this before!
- Bug in QEMU UART emulation which was fixed about a decade ago: If the txfifo becomes full, we don't get an interrupt telling us when it's no longer full.
- Workaround from EC2 many years ago: `hw.broken_txfifo="1"`.

Serial console input

- FreeBSD's serial console couldn't read any input from Firecracker.
- In fact, Firecracker never read any characters from its attached terminal.
- During UART probing, FreeBSD fills its UART input buffer in order to measure its size.
- FreeBSD then writes to the UART FIFO Control Register asking for the input buffer to be flushed.
- Firecracker doesn't implement the UART FCR!
- Workaround: If the UART RXFIFO is not empty after we ask for it to be flushed, read (and discard) bytes until it empties.

Finding devices

- At this point we can boot FreeBSD with a memory disk compiled into the kernel, and interact with it via a serial console.
- ... but that's all we can do because we don't have any devices yet.
- Firecracker exposes devices as `mmio` Virtio devices.
- Step 1: Add `device virtio_mmio` to FIRECRACKER kernel config.

Finding devices

- FreeBSD expects `mmio` Virtio devices to be discovered via FDT.
- Firecracker exposes device metadata via the kernel command line instead:
  `virtio_mmio.device=4K@0x1001e000:5`
  - ... means "a memory-mapped I/O Virtio device occupies 4k of address space starting at `0x1001e000` and uses IRQ 5".
- Step 2: Add `vtmmio_cmdline.c` which can parse such directives and add `virtio_mmio` device nodes.

Finding devices

- FreeBSD parses the kernel command line into environment variables.
- If you have multiple Virtio devices, Firecracker exposes device metadata using `virtio_mmio.device` multiple times.

```
virtio_mmio.device=4K@0x1001e000:5
virtio_mmio.device=4K@0x1001f000:6
virtio_mmio.device=4K@0x10020000:7
```

- Step 3: Rewrite the processing of the early kernel environment (command line and any static values compiled into the kernel) to handle duplicate variables by appending _%d suffixes:

```
virtio_mmio.device=4K@0x1001e000:5
virtio_mmio.device_1=4K@0x1001f000:6
virtio_mmio.device_2=4K@0x10020000:7
```

Unaligned I/Os

- FreeBSD runs fine until you try to `fsck`, at which point the kernel panics.
- It turns out that `fsck` is one of very few things which causes non-page-aligned disk I/Os.
- The Firecracker Virtio blk backend only supports a single segment of data.
- If our buffer is unaligned, we need to bounce the data.
- Fix: Rewrite `virtio_blk` to use `busdma`.

virtio_blk busdma headaches

- I broke dumping, because dumping synthesizes new request structures rather than using structures I prepared for use with `busdma`.
- I broke dumping a second time, because (unlike `x86`) not all architectures accept a `map` argument of `NULL`.
- I broke Virtio on powerpc, because I assumed that `bus` addresses and `physical` addresses were the same thing.
- I think everything is working now...

Current status

- FreeBSD `amd64` `FIRECRACKER` kernel configuration boots in a patched version of the Firecracker VMM.
- Serial console, network, and disk devices work.
- `virtio-vsock` not implemented yet.
- No support for `arm64` yet.
- FreeBSD boots very fast!

Optimizations

- Firecracker is a great environment for noticing performance overhead!
- Having no physical hardware and only minimal devices removes a lot of noise.
  1. Launch FreeBSD VM.
  2. Watch the boot process scroll past in the console.
  3. Tell FreeBSD to "reboot".
  4. Watch shutdown process scroll past in the console.
- After picking low-hanging fruit, use TSLOG.

Shutdown optimizations

- Shutdown was surprisingly slow!
- Issue 1:
  `/* wait 1 sec for printf's to complete and be read */`
- Not very useful since people usually already know that they're rebooting the system!
- Fix 1: Add `kern.reboot_wait_time sysctl`.
- Issue 2: `DELAY(1000000)` in `x86 cpu_reset`.
- BSP was delaying to make sure that APs had shut down, but we now wait for them to set a flag saying that they had stopped.
- Fix 2:
  ```
  cpu_machdep.c | 2 --
  1 file changed, 2 deletions(-)
  ```

Memory optimization

- When booting FreeBSD on a VM with 128 MB of RAM, I found that it was surprisingly unusable.
- After some investigation it turned out that 32 MB of RAM was being reserved by `bus_dma` for bouncing pages.
- In `bus_dma_tag_create`, we were reserving enough bounce pages to bounce an entire `maxsize` request.
- Since pages used for bouncing are typically not contiguous, reserving more than `nsegments` pages is pointless since we won't be able to use them anyway.
- Fix: `x86/busdma: Limit reserved pages if low nsegs`.
- Reduces `bus_dma` memory reservation from 32 MB to 512 kB.

Boot speedup: Entropy seeding

- `random: randomdev_wait_until_seeded unblock wait`
- FreeBSD typically obtains significant entropy from hardware devices, but in a VM we don't have any.
- FreeBSD can also obtain entropy from the UEFI boot loader, but under Firecracker we don't have one.
- On `x86` we make use of the `RDRAND` instruction as a further source of entropy.
- Unfortunately the amount of entropy we asked `RDRAND` for wasn't enough to fill the entropy pool...
- Fix: If Fortuna is not yet "seeded", ask "pollable" entropy sources (e.g. `RDRAND`) for more entropy at once.
- Speeds up the boot process by 2.3 seconds.

Boot speedup: `rc.d/hostid`

- The first time a FreeBSD system boots, it records a host ID.
- When possible, FreeBSD uses `smbios.system.uuid`.
- If that isn't valid, we fall back to generating a random UUID in software.
- `hostid: unable to figure out a UUID from DMI data, generating a new one`
- ... and sleep for 2 seconds to give users a chance to read the warning.
- Firecracker has no SMBIOS.
- Fix: Skip the warning and sleep if we have no hardware UUID.
- Speeds up the boot process by 2 seconds.

Boot speedup: IPv6 DAD on lo0

- IPv6 mandates that systems wait for "Duplicate Address Detection".
- After bringing interfaces up, we wait to see if any other system is using the IPv6 address we picked.
- We did this in `rc.d/netif` if any interfaces had IPv6 enabled.
- We always have IPv6 enabled on `lo0`.
- Fix: Only wait for IPv6 DAD if we have IPv6 enabled on an interface other than `lo0`.
- Speeds up the boot process by 2 seconds.

Boot speedup: `lapic_init`

- In `lapic_init` we calibrate how long `lapic_read_icr_lo` takes to execute.
- This is used by `lapic_ipi_wait` when waiting for IPIs.
- In Firecracker we were spending 10 ms in `lapic_init`.
- Fix:

```
-#define LOOPS 100000
+#define LOOPS 1000
```

- Speeds up the boot process by 10 ms.

Boot speedup: ns8250_drain

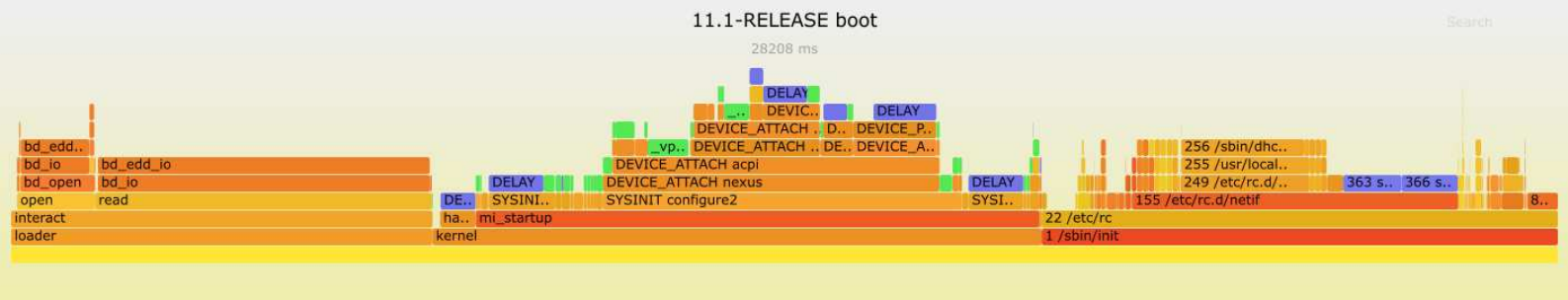
- UARTs in VMs often have fictitious baud rates.
- In ns8250_drain we call DELAY after reading each character.
- In a VM this is unnecessary since the next character becomes available as soon as the previous one is read.
- Fix: Keep reading characters as long as LSR_RXRDY is asserted.
- Speeds up the boot process by 27 ms.

Firecracker speedup: 0x40000010 CPUID leaf

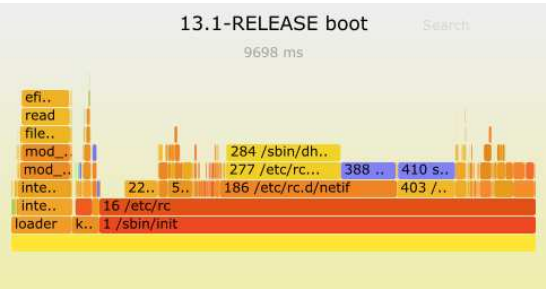
- The 0x40000010 CPUID leaf is used by hypervisors to tell a guest OS the frequencies of the TSC and local APIC timer.
- This allows the kernel to avoid spending time on clocks calibration.
- Implemented by VMWare, QEMU, and EC2, but not Firecracker!
- Fix: Implement the CPUID leaf in Firecracker.
- Speeds up the boot process by 20 ms.
- Also speeds up Linux!

Flame Charts!

Amazon EC2 c5.xlarge instance, 11.1-RELEASE:



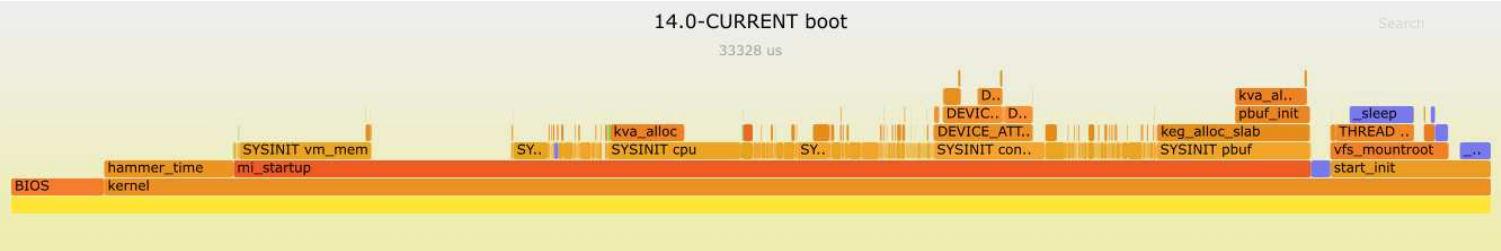
Amazon EC2 c5.xlarge instance, 13.1-RELEASE:



Firecracker, 128 MB RAM, 1 CPU, 14.0-CURRENT:

Flame Chart — kernel boot

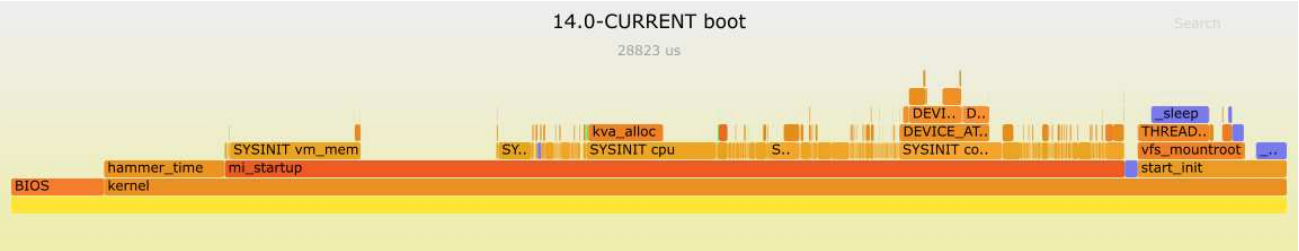
Firecracker, 128 MB RAM, 1 CPU, 14.0-CURRENT:



- Time before entering userland: 33.382 ms.
- 4.184 ms spent in `keg_alloc_slab`.
- 3.181 ms spent in `kva_alloc`.
- 2.765 ms spent in `SYSINIT vm_mem`.

Flame Chart — kernel boot

Firecracker, 128 MB RAM, 1 CPU, 14.0-CURRENT:



- Time before entering userland: ~~33.382~~ 28.823 ms.
- ~~4.184~~ 2.566 ms spent in `keg_alloc_slab`.
- ~~3.181~~ 1.884 ms spent in `kva_alloc`.
- 2.765 ms spent in `SYSINIT vm_mem`.

TODO

- Add FIRECRACKER kernels to release engineering builds.
- Finish separating out "Xen" bits from "PVH booting" bits.
- Remove PCI, ACPI, and XEN from FIRECRACKER config.
- Attach `virtio_mmio` devices to a `virtio_cmdline` "bus" instead of attaching them to `nexus0`?
- Implement `virtio_vsock`?
- Get PVH support patches merged to Firecracker.
- Fix Firecracker UART to properly interrupt when txfifo empties.
- Implement the UART FIFO Control Register in Firecracker.
- Convince Amazon to support FreeBSD in AWS Lambda.
- Can we port Firecracker to run on FreeBSD?

Thanks

- I found the problems but other FreeBSD developers fixed most of them:

  Bryan Drewery
  Ed Maste
  Jessica Clarke
  John Baldwin
  Kyle Evans
  Mark Johnston
  Roger Pau Monné
  Warner Losh

Availability

- FIRECRACKER kernel configuration in FreeBSD 14.0-CURRENT.
- Patched Firecracker is available in GitHub:
  `https://github.com/cperciva/firecracker`
  branch `pvh-v3-tsc`
- Coming soon to upstream Firecracker (hopefully):
  `https://github.com/firecracker-microvm/firecracker`

Questions?