Linux Kernel on RISC-V: Where do we stand?

Atish Patra, Principal R&D Engineer

Damien Le Moal, Director, System Software Group
Overview

• Software ecosystem status overview
  – Development toolchain
  – Linux distributions
  – Linux kernel

• What’s next?

• Our learnings

• Contribution process

• Summary
Why Linux?

**TOP 1M WEBSITES**
- Linux: 96%
- Others: 4%

**x86 SERVER**
- Linux: 50%
- Others: 50%

**MOBILE DEVICES**
- Android: 77%
- Others: 23%

**IOT DEVICES**
- Linux: 72%
- Others: 28%
Software Ecosystem Overview

User Space
- Applications
- Glibc, core libraries

Linux Kernel
- CPU extensions device drivers
- Memory Management (MMU, IOMMU)
- Core Device drivers (CPU management, interrupt management, ...)

Linux Distribution
- Supervisor Binary Interface (SBI)
- Bootloader (BBL)
- RISCV CPU
Software Ecosystem Growth

March 17
Binutils

May 17
GCC

Nov 17
4.15 Linux Kernel

Jan 18
Glibc

March 18
QEMU

March 18
Debian port start

Feb 18
Fedora bootstrap done

©2018 Western Digital Corporation or its affiliates. All rights reserved.
Development toolchain

• Almost Complete Development Environment
  – GCC 7.3.1
  – Perl 5.26
  – Python 2 and 3
  – Git
  – Systemd
  – LLVM
Linux Distributions: Fedora

- 1st Linux distro available on RISC-V
- Fedora 29 available
- The koji build farm
  - Contains 3 boards + 6 QEMU VMs
  - 3000+ builds a week, 500+ a day
  - The best was 85+% success rate for a week
- 22347 number of packages are built
Linux Distributions: Debian

- More than 80% Debian packages are built for RISC-V
- Higher than Itanium!!
- More than 9k of ~13k arch-dependent packages built
Demonstration (Thanks to Palmer)

- So what Can I do on a RISC-V desktop?
  - Browsing
  - 3D gaming
  - Chat/Tweet
  - Crypto currency mining ???

©2018 Western Digital Corporation or its affiliates. All rights reserved.
Kernel Status: Supervisor Binary Interface (SBI)

• What is SBI?
  – OS interface for Supervisor Execution Environment (SEE)
  – Equivalent to microcode implementation
  – Implemented by Berkely Boot Loader (bbl)
  – Provides cleaner interface for Supervisor OS (i.e. Linux)

• Interface functions
  – Set Clock events
  – Send/Clear Inter Processor Interrupts (IPI)
  – Serial Console Get/Update
  – Remote Memory barrier (aka fence)
Kernel Status: Core Features

- UMP/SMP Support
- Interrupt Controller
- Clock driver
- Virtual memory support
- Serial Driver
- System call/trap handling
- Kernel module support
- CPU hotplug support
Kernel Status: RISC-V Boot Process

- No dedicated boot CPU
- A lottery based mechanism to select the boot CPU
- Non-boot CPUs boot in random order
- ~100 lines of assembly instructions
- All device initialization depends on device-tree
- Device tree is read from ROM & passed to kernel by BBL
Kernel Status: Interrupt Management

[Diagram showing interrupt management flow with labels for Hardware and Kernel, including components like External Interrupt, Timer Interrupt, Software Interrupt, and Local Interrupt.]
Kernel Status: Timer Management

• One timer per hart
• Per CPU based clock event
• Clockevents are initialized via CPU hotplug during SMP bring up
• One shot timer
• Every clock event reprograms next one via SBI call
• All the timers across all the harts are synchronized within one tick of each other
• Tickless kernel (NOHZ) support is also enabled
Kernel Status: CPU Hotplug

```
# cat /proc/cpuinfo
hart: 1
isa: rv64imafdc
mmu: sv39
urarch: sifive,rocket0

# cat /proc/cpuinfo
hart: 2
isa: rv64imafdc
mmu: sv39
urarch: sifive,rocket0

# cat /proc/cpuinfo
hart: 3
isa: rv64imafdc
mmu: sv39
urarch: sifive,rocket0

# cat /proc/cpuinfo
hart: 4
isa: rv64imafdc
mmu: sv39
urarch: sifive,rocket0
```
Kernel Status: Tracing/Debugging

- Ftrace
  - Normal trace
  - Function graph tracing
  - Function profiling
  - Filtering
  - No kprobes
- Perf
  - Instructions count
  - Cycle count
- GDB port is in progress
What’s next – User space

• High performance Java support
  – OpenJDK port by J extension group going on
  – Jikes RVM going on. Not yet public

• In progress
  – Rust
  – GO, Clang

• JavaScript
  – V8
  – Node.js

• Others
  – Dart, Ruby
  – GTK, OpenCV
What’s next – Core Linux Kernel

- Independent boot loader (uboot or coreboot)
- SBI extension for power management/virtualization
- CPU topology
- Multi-level interrupt controller through irq domains (in progress)
- CPU power management
- Context tracking for NO_HZ_FULL
- Precise IRQ time accounting
- IRQ stats update for Timer/IPI interrupt
What’s next – Core Linux Kernel

• Specification required
  – IOMMU
  – Performance Monitor counters
  – Virtualization support

• Not useful at this time due to lack of hardware
  – Numa
  – Memory hotplug
  – Huge pages
What’s next – Tracing/Debugging

• Kexec
• Kdump
• Lockdep
• Kasan
• Kprobes
• kernel function-return probes
• KGDB
• BPF
Our Learnings

• QEMU verification is very handy
• NFS & chroot in QEMU brings up the Fedora/Debian without any costly hardware!!
• Remote gdb debugging possible in QEMU
• Print device tree in bbl
• Use Ftrace to analyze kernel issues
• Print to debug kernel
• Dumping dmesg in QEMU
• Inspecting registers in QEMU
Contribution Process

• RISC-V port available in Mainline Linux since v4.15
• The RISC-V development tree
  – kernel.org/palmer/riscv-linux.git
  – master: a copy of Linus’ master, up to the latest RC.
  – for-linus : RISC-V branch that’s pulled into Linus’ master branch
  – for-next : RISC-V branch that’s pulled into linux-next
  – riscv-all : RISC-V integration branch
• Patches should be sent to
  – linux-riscv@lists.infradead.org
• IRC Chatrooms on freenode
  – #linux-riscv
  – #riscv
Atish Patra
atish.patra@wdc.com