RISC-V Patchwork CI

Björn Töpel, Rivos
Conor Dooley, Microchip
Rationale

● Pre-merge hook
● “Is this patch good enough?”
● Remove burden from maintainers
History

● Patchwork CI – All subsystems have their own
● Forked netdev’s NIPA infra
  ○ Python, mostly hacked by Jakub Kicinski
● Adapted for RISC-V
● Hosted by Conor/Microchip
Pain points

- Black box (-ish)
- One man army (Conor)
  - Administration – “No one wants to be the CI person”
  - Changes
- Enter RISE...
PW CI NG

- RISE/Google provided VMs
- Move to Github
- netdev BPF! Fork Meta’s KPD (Kernel Patches Daemon)
- (Hopefully)
  - Easier to maintain
  - Encourage community CI hacking
  - Easier to scale
- https://github.com/linux-riscv
for i in series 1

PW (REST API)

riscv trees + CI

GH linux-riscv.git

RISE Google cloud

KPD

Runner 1

Runner 2
for i in series 2

PW (REST API)

riscv trees + CI

GH linux-riscv.git

RISE Google cloud

KPD

Runner 1

Runner 2
for i in series 3

PW (REST API)
riscv trees + CI
GH linux-riscv.git

RISE Google cloud

Runner 1
KPD
Runner 2
for i in series 4

- PW (REST API)
- riscv trees + CI
- GH linux-riscv.git

RISE Google cloud
- Runner 1
- KPD
- Runner 2
for i in series 5

- PW (REST API)
- riscv trees + CI
- GH linux-riscv.git

RISE Google cloud

Runner 1

KPD

Runner 2

✅
for i in series 6

- PW (REST API)
- riscv trees + CI
- GH linux-riscv.git

RISE Google cloud

Runner 1

KPD

Runner 2
for i in series 7

- PW (REST API)
- riscv trees + CI
- GH linux-riscv.git
KPD

- Meta’s KPD forked: [https://github.com/linux-riscv/kernel-patches-daemon](https://github.com/linux-riscv/kernel-patches-daemon)
- Synchronizes with upstream: for-next, fixes
- Applies series from PW onto upstream
  - Actual patches
- Creates GH pull requests
- Scrapes results from GH and publishes to PW (RV specific)
CI “.github”

- Direct port of NIPA-RV
- Per-patch builds/tests
- Bash + YAML glue (“easy to run locally”)
- .github/scripts/patches

build_rv32_defconfig.sh
build_rv64_clang_allmodconfig.sh
build_rv64_gcc_allmodconfig.sh
build_rv64_nommu_k210_defconfig.sh
build_rv64_nommu_virt_defconfig.sh
checkpatch.sh
dtb_warn_rv64.sh
header_inline.sh
kdoc.sh
module_param.sh
verify Fixes.sh
verify_signedoff.sh
Github Runner

- Docker based build: [https://github.com/linux-riscv/docker](https://github.com/linux-riscv/docker)
  - arm64 cross-builder
  - kernel.org GCC/LLVM
  - Bleeding edge dt-schema
  - tuxmake for building kernels
- “Ephemeral runners”
## Grokking traffic lights

<table>
<thead>
<tr>
<th>Patch</th>
<th>Series</th>
<th>AR/F</th>
<th>SWIF</th>
<th>Date</th>
<th>Submitter</th>
<th>Delegate</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>[v7.3/3] riscv: Add tests for riscv module loading</td>
<td>riscv: Add remaining module relocations and tests</td>
<td>- - -</td>
<td>10 1</td>
<td>2023-10-31</td>
<td>Charlie Jenkins</td>
<td>New</td>
<td></td>
</tr>
<tr>
<td>[v7.2/3] riscv: Add remaining module relocations</td>
<td>riscv: Add remaining module relocations and tests</td>
<td>- - -</td>
<td>10 8</td>
<td>2023-10-31</td>
<td>Charlie Jenkins</td>
<td>New</td>
<td></td>
</tr>
<tr>
<td>[v7.1/3] riscv: Avoid unaligned access when relocating modules</td>
<td>riscv: Add remaining module relocations and tests</td>
<td>- - -</td>
<td>12 1</td>
<td>2023-10-31</td>
<td>Charlie Jenkins</td>
<td>New</td>
<td></td>
</tr>
</tbody>
</table>
Grokking GH PRs
“Demo”

https://github.com/linux-riscv
Resources

- RISC-V GH landing page: https://github.com/linux-riscv/
- CI WIKI: https://github.com/linux-riscv/github-ci/wiki
- RISE: https://wiki.riseproject.dev/display/HOME/Developer+Infrastructure+WG
- BPF CI slides from LSFMM ‘22 ‘23:
  http://vger.kernel.org/bpfconf2023_material/BPF_CI_a_year_later.pdf
Next steps

- Add per-series tests ~ "Palmer’s release tests"
  - 144 build configs (rv32, rv64, GCC, LLVM)
  - 216 qemu “boots” (buildroot-rv32, ubuntu, alpine)
  - Each boot (DT, ACPI, UEFI-uboot, UEFI-EDK2, non-UEFI)
Next steps

- CI performance
  - Utilize all cores (compilation vs linking vs booting)
  - On-demand Runners (web_hooks)?
  - Spot/preemptible Runners?
Discussion

- Next steps build/boot
  - What’s missing? Email contributors?
  - Too much?
- UI?
- Overlap with kernel-ci et al?
- Native builds?
- Native boots! (Boards...)
- PRs re-triggered on base change. Too much?