What is LKFT and how does it improve the Linux kernel overall quality

Dan Rue
Kernel Validation Team Lead
dan.rue@linaro.org
@mndrue
Obligatory

- From Minnesota, USA, near Minneapolis
- At Linaro for 1.5 years (3rd Connect, 1st talk!)
- DevOps guy.. Whatever that means
- Like python, golang, open source
- Prefer red hats and dark beer

See what I did there
Intro: LKFT Topics

- Branches
- Architecture
- Hardware
- LAVA Lab
- Tests and Coverage
- Triage Process
- Reporting
- Kernel CI
- 6 Month Recap (fast!)
Intro: Definitions

Linux Kernel Functional Testing

- Under test: Upstream Linux Kernel
  - Not (necessarily) boards
  - Not (necessarily) builds
  - Not (necessarily) boots
- Functional Testing
  - No performance testing
- Focus on detecting regressions
- Focus on stable kernel branches
Intro: Scope & Caveat

- LKFT is a framework that is used to test a variety of kernel trees on a variety of user spaces and hardware
- This talk will focus on testing stable, mainline, and next trees using OpenEmbedded
Linux Kernel Branches

- next
- mainline
  - 4.19 - expected LTS
  - 4.18 - current stable
  - 4.14 - EOL Jan, 2020
  - 4.9 - EOL Jan, 2023
  - 4.4 - EOL Feb, 2022

- .. and now, developer trees based on above branches (see Daniel’s talk)
* Actually, we build arm64 once for each board.
Hardware - Primary

Primary Hardware Requirements
- Reliable
- Performant
- Upstream Support, especially LTS

See [https://lkft.linaro.org/faq/](https://lkft.linaro.org/faq/) for detailed hardware criteria
Hardware - Community

Community Hardware Considerations
- Inexpensive
- Highly Available
- Upstream Support, especially LTS

DragonBoard 410c* (64bit)

BeagleBone Black* (32bit)

* Tentative
# LAVA Lab: Devices

## Device Type Overview

<table>
<thead>
<tr>
<th>Name</th>
<th>Idle ↓↑</th>
<th>Offline ↓↑</th>
<th>Busy ↓↑</th>
<th>Restricted ↓↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>dragonboard-410c</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>hi6220-hikey-r2</td>
<td>6</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>juno-r2</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>qemu</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x15</td>
<td>4</td>
<td>9</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>x86</td>
<td>1</td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

[https://lkft.validation.linaro.org/scheduler/](https://lkft.validation.linaro.org/scheduler/)
LAVA Lab: 48h Queue Depth
Tests and Coverage - Full

*Over 20,000 tests per kernel!*

- 10 Environments
  - dragonboard-410c - arm64
  - hi6220-hikey - arm64
  - i386
  - junor2 - arm64
  - qemu_arm
  - qemu_arm64
  - qemu_i386
  - qemu_x86_64
  - x15 - arm
  - x86_64

- Test runs per environment
  - Kselftest - ~50-100 tests
  - Ltp - ~3000 tests
  - Libhugetlbfs - ~90 tests
Dealing with test results is hard
Triage Process: Skips and Known Issues

- First approach: Try to run everything.
  - Consequence: Boards crash; tests hang and cause timeouts.
Triage Process: Skips and Known Issues

- First approach: Try to run everything.
  - Consequence: Boards crash; tests hang and cause timeouts.

- Second approach: Try to run everything, but skip tests that cause the test run to fail
  - Consequence: Lots of failures to wade through
Triage Process: Skips and Known Issues

- First approach: Try to run everything.
  - Consequence: Boards crash; tests hang and cause timeouts.
- Second approach: Try to run everything, but skip tests that cause the test run to fail
  - Consequence: Lots of failures to wade through
- Third approach: Compare previous run to current and detect ‘regressions’ and ‘fixes’
  - Consequence: Lots of tests fail intermittently.
Triage Process: Skips and Known Issues

- First approach: Try to run everything.
  - Consequence: Boards crash; tests hang and cause timeouts.
- Second approach: Try to run everything, but skip tests that cause the test run to fail.
  - Consequence: Lots of failures to wade through.
- Third approach: Compare previous run to current and detect ‘regressions’ and ‘fixes’.
  - Consequence: Lots of tests fail *intermittently*. Lack of good baselines.
- Fourth approach: Skip failing tests to make the data more clear.
  - Consequence: No more ‘fixes’. Hiding too much data.
Triage Process: Skips and Known Issues

- First approach: Try to run everything.
  - Consequence: Boards crash; tests hang and cause timeouts.
- Second approach: Try to run everything, but skip tests that cause the test run to fail
  - Consequence: Lots of failures to wade through
- Third approach: Compare previous run to current and detect ‘regressions’ and ‘fixes’
  - Consequence: Lots of tests fail *intermittently*. Lack of good baselines.
- Fourth approach: Skip failing tests to make the data more clear.
  - Consequence: No more ‘fixes’. Hiding too much data.
- Fifth approach (Current): Triage and annotate failures
  - Cons: High touch, not automated
  - Pros:
    - Any known failure that passes is a ‘fix’ (unless ‘intermittent’)
    - Any new failure is a regression (unless ‘intermittent’)
Triage Process: Bug Classification

Status

- **UNCONFIRMED** - New bugs, not yet assigned.
- **CONFIRMED** - Assigned for triaging to determine root cause, and report/escalate/fix accordingly.
- **IN_PROGRESS** - Root cause has been identified and being worked on.
- **RESOLVED** - Root cause fixed. i.e. patch sent
- **VERIFIED** - Fix has been accepted, backported as necessary, and test can be removed from skiplist/known issues list.

https://bugs.linaro.org - Product ‘Kernel Functional Testing’
Triage Process: Bug Classification

Component

- **Kernel** - Actual kernel issues, excluding issues under tools/testing/selftests/ (kseltest).
- **General** - Bugs that don’t fit any other component. Examples include build issues, root filesystem issues, etc.
- **kseltest** - Issues that need to be fixed in kseltest.
- **Linux Test Project (LTP)** - Issues that need to be fixed in LTP.

[https://bugs.linaro.org](https://bugs.linaro.org) - Product ‘Kernel Functional Testing’
Triage Process: Bugs over time

Status Counts for Kernel Functional Testing

Source @ bugs.linaro.org
Triage Process: Stable
Triage Process: Stable

- Stable Release Candidate (RC) Log @ https://lkft.linaro.org/logs/rclog/
- Service Level Agreement (SLA) for stable results: 48h turnaround
- Since Jan 1, 2018
  - Total RCs: 221
    - SLA >48h: 4
    - SLA <48h: 35
    - SLA <24h: 151
    - SLA <8h: 31
    - SLA met 98.2% of the time
Reporting: email

- SQUAD automatically generates reports once a set of tests finishes.
- Fully templatized for ad-hoc customization
- Garbage in garbage out can be an issue (see previous slides)
  - **Bad test data will create email reports that get ignored**
- Augmented with API-generated reporting (from Linaro/lkft-tools)
  - Set custom baseline, generate for arbitrary build
Reporting: qa-reports.linaro.org

### Reporting: qa-reports.linaro.org

**Compare (kselftest/net_reuseport_bpf_numa)**

#### lkft/linux-stable-rc-4.18-oe

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>dragonboard-410c-arm64</th>
<th>hi6220-hikey-arm64</th>
<th>junor2-arm64</th>
<th>qemu_arm</th>
<th>qemu_arm64</th>
<th>qemu_x86_64</th>
<th>x15_arm</th>
<th>x86_64</th>
<th>i386</th>
<th>qemu_i386</th>
</tr>
</thead>
<tbody>
<tr>
<td>v4.18.7-198-gae285c2014a</td>
<td>2018-09-13T16:20:50.193383Z</td>
<td>pass</td>
<td>pass</td>
<td>n/a</td>
<td>fail</td>
<td>pass</td>
<td>pass</td>
<td>fail</td>
<td>pass</td>
<td>skip</td>
<td>skip</td>
</tr>
<tr>
<td>v4.18.7-191-g914a2a9e27a</td>
<td>2018-09-12T20:53:52.619989Z</td>
<td>pass</td>
<td>pass</td>
<td>n/a</td>
<td>fail</td>
<td>pass</td>
<td>pass</td>
<td>fail</td>
<td>pass</td>
<td>skip</td>
<td>skip</td>
</tr>
<tr>
<td>v4.18.7-154-g1675305fe5e7</td>
<td>2018-09-11T15:14:30.265540Z</td>
<td>pass</td>
<td>pass</td>
<td>n/a</td>
<td>fail</td>
<td>pass</td>
<td>pass</td>
<td>fail</td>
<td>pass</td>
<td>skip</td>
<td>skip</td>
</tr>
</tbody>
</table>

[https://qa-reports.linaro.org/](https://qa-reports.linaro.org/)  
/comparetest/...
Reporting: lkft.linaro.org

Linaro Connect
YVR18

Test Results

LKFT Build Status
The LKFT build uses OpenEmbedded to build a userspace image, along with the kernel, for each board and branch combination under test.

<table>
<thead>
<tr>
<th>Board</th>
<th>4.4</th>
<th>4.9</th>
<th>4.14</th>
<th>4.18</th>
<th>mainline</th>
<th>linux-next</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hikey arm64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build failing</td>
</tr>
<tr>
<td>X15 arm32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>build failing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
</tr>
<tr>
<td>Juno arm64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build failing</td>
</tr>
<tr>
<td>DragonBoard 410c arm64</td>
<td>Not supported</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build failing</td>
</tr>
<tr>
<td>Intel Server i386</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
</tr>
<tr>
<td>Intel Server x86_64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
<td>build passing</td>
</tr>
</tbody>
</table>

https://lkft.linaro.org/
Kernel CI

- Build and Boot testing for upstream kernel done really well by [https://kernelci.org/](https://kernelci.org/)
- Kernel CI has begun implementing functional testing
- Currently exploring ways to improve kernelci based on our work in LKFT
- Follow kernelci
  - [https://kernelci.org/](https://kernelci.org/)
  - [https://groups.io/g/kernelci](https://groups.io/g/kernelci)
  - freenode/#kernelci
6 Month Recap (fast!)

- qa-reports upgraded from SQUAD 0.40 to 0.50 (changelog)
- Branches: 4.15 removed, 4.16/4.17 added and removed
- QEMU arm64 and arm32 added
- LTP CVE tests added
- LTP upgraded to 20180515
- Kselftest upgraded to 4.16, 4.17, 4.18
- LTP Open Posix tests added
- OpenEmbedded upgraded from Morty to Rocko
- Fast sanity tests implemented
- i386 support added

https://lkft.linaro.org/logs/changelog/
https://lkft.linaro.org
lkft@linaro.org
freenode/#linaro-lkft
Dan Rue
dan.rue@linaro.org
@mndrue