Linaro’s LKFT

Linaro created the Linux Kernel Functional Test (LKFT) framework to:

- Functional test the Linux kernel under full user-space images
- Test on a variety of hardware
- Identify regressions against previous releases or baselines
- Report the results upstream
- Provide detailed results via a dashboard
**LKFT Mission**

- Identify Linux kernel functional regressions

- Shorten derivative Linux kernel release intervals e.g., LTS -> android common kernel -> SoC trees

- Increase the confidence of upstream Linux kernel engineers in the quality of their releases  
  - Greg KH told me that LKFT has already shortened his LTS rc-to-release process by ‘half-a-day’

- Increase the confidence of downstream adopters (e.g., phone vendors) that deploying updated kernels to their devices will not regress functionality or security of their products
LKFT Goal

LKFT will give downstream hardware vendors the confidence to more frequently update the Linux kernel that runs on their devices.

Consumers will benefit from bug and security updates.
**Operational LKFT**

- **[Operational]** Linaro Project Sharp
  - Linux LTS and Android-common (LTS derived) trees in support of Google Project Treble

- **[Operational]** Linux Community Upstream Testing
  - Linus’s mainline, stable-rc, stable-rc, linux-next trees

- **[Operational]** ARM SoC testing of LTS/LCR/RPK derived trees
  - Qualcomm db410c

- **[Development In Progress]** Linaro Kernel Development Testing

- **[Development In Progress]** LSK & LSK RT Regression Testing
LKFT Collaboration

● How can the community get involved?
  ○ Consume the results of LKFT regression testing
  ○ We want to improve the reporting, testing coverage, and reporting interface based on feedback.
    ■ We are actively working on tailoring the interface of our tools to streamline the workflow of kernel engineer consumers

● How can non-Member hardware be integrated?
  ○ LKFT is implemented by Linaro as a privilege of Linaro membership
  ○ Any non-member hardware that is integrated is at Linaro’s discretion

● How can Members get involved?
  ○ Linaro can administer some member devices in LKFT directly
  ○ Member’s engineers can help with device specific testing triage
  ○ But many member devices are a candidate for LKFT Remote Labs
LKFT Remote Labs

A method to extend the reach of Linux kernel functional testing to Linaro member SoCs in a way that can:

- Accommodate member privacy/confidentiality requirements
- Address device complexity concerns of Linaro engineers and respect Linaro staff constraints

This presentation will describe the proposed capabilities, deficiencies, and compelling arguments for adoption of remote LKFT labs by Linaro members.
LKFT Overview
LKFT - Description (Scheduling Builds)

**Builds scheduling** is done by the [Linaro Jenkins CI](https://www.linaro.org) instance.

Collectively the LKFT framework watches targeted Linux kernel git repositories for changes and then automatically builds Linux kernel images that it combines with AOSP and OpenEmbedded images. The result is a system image that is ready to be flashed to a device for functional testing.
**LKFT - Description (Scheduling Jobs)**

**Job scheduling** starts when the Linaro **Jenkins** continuous-integration system informs the Linaro **SQUAD** instance that new system images are available for testing.

The Linaro **SQUAD** instance is the authority that schedules test jobs by submitting LAVA job definitions to Linaro’s **LKFT LAVA** instance.
LKFT - Description (Test Execution)

**Test Execution** in the LKFT framework is dispatched by Linaro’s LKFT LAVA instance once it has received job definitions from SQUAD. LAVA dispatches test jobs on its devices, downloads system images from [http://snapshots.linaro.org](http://snapshots.linaro.org), flashes the system images, boots the devices, executes tests, and reports the results to the LAVA database.

The LAVA UI presents the executing job, and device information.
Reporting is provided by Linaro’s SQUAD dashboard. SQUAD communicates with the LAVA Master scheduler and watches for test job completion. SQUAD not only visualizes the validation results it collects from the LAVA database but it also generates email reports.

https://qa-reports.linaro.org/lkft
lkft.linaro.org

https://lkft.linaro.org is a website for kernel engineers, business partners, and managers to get up-to-date information on functional test results against the latest commits to a variety of Linux kernel source trees.

This website is a work in progress at the time of this writing
Test results triage is performed by Linaro triage teams. They interpret results of automated testing and marshal the issues to the correct place:

- Notify relevant mailing lists of identified regressions
- Differentiate and report infrastructure failures for correction
- Identify Linux kernel bugs
- Identify test-suite bugs
Hardware in Linaro’s LKFT Lab

- HiSilicon HiKey 6220 (96board)
- TI AM572x (BB-X15)
- Arm Juno
- x86_64
- Qualcomm db410c
- ST B2260
- Qemu
Hardware Selection Criteria for Linaro’s LKFT Lab

The model LKFT device has the following characteristics:

- Structurally stable under 24/7 CPU load and memory pressure
- Upstream Kernel and Userspace Support (OE and AOSP)
- Available in Linux LTS trees
- Easy to enable in LAVA
- Software controlled power-cycling
- Sufficient RAM for functional testing
- Stable firmware
- Reasonably affordable and available in quantity
- Can support secondary (non-eMMC/Flash, non-SD Card) storage

Not all devices in Linaro’s LKFT lab satisfy these requirements
Why Such Strict Selection Criteria in Linaro

- Reliability reduces necessity to replace devices and admin intervention
- Custom hardware work arounds increase complexity of triage and result in cannot-fix failures.
- Management cost of deploying devices at scale must be minimal
- Unbricking boards is time consuming as it generally requires unracking
- Triage time must be minimal for LKFT to be useful upstream
  - Maintainers need to know that release-candidates have regressions as soon as possible
- Linaro can’t maintain custom trees that need merge pushes, as this introduces uncertainty, increases maintenance costs, and reduces time to execute after upstream pushes.
- Flaky hardware, and unknown hardware failures, that require frequent resubmission increases response time of validation beyond useful time period.
- Under performing hardware takes too long to execute full test runs.
- Closed firmware means some bugs result in cant-fix failures.
LKFT Expansion Opportunities

- Increase test coverage
- Increase tree coverage
- Increase hardware coverage
- Improve tools and reporting
Constraining Engineering Cost

Linaro must critically consider the addition of any hardware platform or kernel tree into LKFT:

- Will the device have a positive impact that exceeds the expense of maintaining it ‘in house’?

- Some choices of hardware & kernel trees are easily implemented and maintained while others are significantly more expensive

- Linaro recognizes that there is still value in the LKFT framework for members and their hardware even if Linaro can’t execute on every member platform

- Some hardware is not yet public but early testing is desirable
Reasons For Member LKFT Remote Labs

The **LKFT Remote Lab** concept is a way to empower members to functionality test their hardware against the Linux kernel with structured frequency using a common framework and reporting system.

- Easy access desktop support for hard-to-rescue targets
- Faster time-to-bringup and staging of new hardware
- Allow members w/ proprietary firmware to test their firmware changes
- Hard-to-automate hardware solutions can be implemented and maintained by members.
- LAVA integration is member’s responsibility, done by the hardware experts
- Member pre-release hardware can be part of LKFT (privately)
- Purpose of member LKFT hardware is to notify members of regressions in their software stack (additions) due to upstream changes, not the other way around.
Safeguarding Member Privacy

SoC git trees can be maintained in private
Device availability can be under access-control
Job dispatch and visibility can be under access-control
Test results and reporting are under access-control
Bugs can be under access-control
Member Maintained Linux Kernel SoC git trees

Members can maintain their SoC git trees and resolve merge conflicts with LTS themselves. Linaro engineers aren’t burdened if the SoC tree merge isn’t clean.

Binary blobs and non-upstreamed code can be tested
What’s It Take? - Partially Hosted

Member Hosts Minimum LAVA

- Cloud accessible git repository of SoC tree derived from LTS, maintained by members
- Member hosted LAVA Dispatcher
- Member hardware w/ device farm behind firewall
- Linaro hosted LAVA Master (jobs and devices can remain private)
- Requires a permissive tunnel between Linaro LAVA Master and member LAVA dispatcher
  - Dispatcher software version must maintain compatibility with LAVA master version.
- Linaro creates builds and images (if possible)
- Member can push results to Squad w/ callback API or Squad can use HTTP polling
  - [https://qa-reports.linaro.org/lkft/](https://qa-reports.linaro.org/lkft/)
  - Results can be kept private.
What’s It Take? - Fully Hosted

Member Hosts Entire LAVA Hardware & Software Stack

- Cloud Accessible Git Repository of SoC tree derived from LTS
  - Members maintain SoC git tree
- Image builds by Linaro Jenkins
- Member local LAVA Master, Dispatcher, Database, & devices
- Doesn’t require VPN access into member LAB
- Requires a tunnel between LAVA Master and Linaro Squad
- Member can push results to Squad w/ callback API
What’s It Take? - Ultra-secure

Member Hosts Entire LAVA Hardware & Software Stack, Git Servers & Builds

- Member local SoC git tree derived from LTS
  - Members maintain SoC git tree of Linux kernel
- Member local LAVA Master, Dispatcher, Database, w/ device farm
- Highest Member Input
- Highest Security
  - Doesn’t require a two-way tunnel
  - Linaro has no access to software, images, or hardware
- Member LAVA master must push results to Squad
  - https://qa-reports.linaro.org/lkft/
LKFT Remote Lab Prototype

- Ultra-secure model
- Test device: HiKey 6220
- Thinkpad T440 running Debian 9.3 as lava-slave dispatcher
- Raspberry Pi3 plus relay module for power/OTG on/off/reboot control
- ORICO USB3.0 Hub with Gigabit Ethernet Converter for:
  - Connection to HiKey’s serial and OTG USB ports
  - Connection to Raspberry Pi3’s Ethernet port to control reply module over ssh
- lava-master running on the cloud: https://chase.lkft.org/scheduler/device/hi6220-hikey-01
- Linaro LKFT qa-reports displaying LKFT remote lab prototype https://qa-reports.linaro.org/staging-lkft/remote-lab-demo/

Come and see Chase’s demo on Demo Friday
Prototype Example Results

https://qa-reports.linaro.org/staging-lkft/remote-lab-demo/

| build-url | http://192.168.11.205:8080/job/openembedded-l4-filesystems/rmstest/driver-stretch-andr5440/
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>git branch</td>
<td>master</td>
</tr>
<tr>
<td>git commit</td>
<td>f1a6f5064484be074929513a5a446a2958be3948</td>
</tr>
<tr>
<td>git describe</td>
<td>v4.16-rc3-97-gf3a6f32d644</td>
</tr>
<tr>
<td>git repo</td>
<td><a href="https://git.linaro.org/openembedded/l4/filesystems">https://git.linaro.org/openembedded/l4/filesystems</a></td>
</tr>
<tr>
<td>kernel-config</td>
<td><a href="http://192.168.11.205:8080/job/openembedded-l4-filesystems/rmstest/driver-stretch-andr5440/config">http://192.168.11.205:8080/job/openembedded-l4-filesystems/rmstest/driver-stretch-andr5440/config</a></td>
</tr>
</tbody>
</table>
Prototype Instructions

Chase Qi has provided detailed instructions on how to set up an LKFT Remote Labs prototype using all three of the described models:

https://github.com/Linaro/lkft-remote-lab

He provides the following instructions:

● Instructions to setup a LAVA master & dispatcher
● Instructions to setup a docker based jenkins with system image build scripts
● Instructions for connecting to Linaro’s LKFT reporting system
● Instruction to setup a power relay module controlled by GPIO for power cycling devices
Interested In LKFT Remote Labs?

Contact Linaro Developer Services
services@linaro.org
Thank You

#HKG18
HKG18 keynotes and videos on: connect.linaro.org
For further information: www.linaro.org
KernelCI Capabilities Compared to LKFT

Why LKFT and not a functional test framework extension of KernelCI?

- At the time LKFT was created KernelCI did not have any aspirations for functional test (or they weren’t public).

- From the beginning LKFT has been focused on functional testing specific kernel trees (that match Linaro’s membership motivations).

- Even now, as support for kselftest is being added to KernelCI, there is minimal filesystem support, so it does not yet match, 1-for-1, the functional test capabilities of LKFT.
LKFT compared with KernelCI

**LKFT**
- Functional testing as a first-order design requirement
- Full userspace
- Functional test coverage
- Limited hardware due to userspace requirements
- Linaro member needs driven
- Linaro member goal driven - Project Sharp, extend LTS, LSK testing, et al.
- Does boot-test limited hardware
- Limited only by Linaro & member development pace

**KernelCI**
- Boot testing as a first-order design requirement
- Minimal userspace
- Boot test quickly
- Larger class of hardware supported
- Community consensus driven
- Linux community goal driven
- Can functional test w/ minimal userspace
- Limited by pace of community consensus
- Open devices only
- Cannot publish results under access control
Why a Linaro service and not a community project

● LKFT requires involved curation and triage by Linaro engineers

● LKFT can be as responsive to changes in LTS trees as necessary due to investment by Linaro and its members

● Having LKFT as a Linaro service can make sure the effort is comprehensive and meets expectations rather than just minimal-effort
Barriers & Concerns to LKFT Remote Labs

There is some complexity in setting up LAVA Master & Dispatcher(s).
How do Member engineering staff reproduce problems that LKFT finds?
How do we support proprietary binary blobs?
How do we support custom flashing tools that can’t integrate with LAVA?