Customs tests in Free Electrons's lab

Florent (Skia) Jacquet

June 7, 2017

Free Electrons

Why custom tests

- Just booting is not enough.
- Need to test some specific components such as USB, SATA, network...
- Also need to test with custom kernel images, or custom rootfs, or both.
- Need to be able to send jobs, on demand, easilly, on the specified boards, while being notified at the end.

Overall architecture

Many tools

- LAVA
- Buildroot
- The custom test tool
- The test suite

LAVA

- `https://www.linaro.org/initiatives/lava/`
- Runs jobs on devices.
  A job define a procedure to run, and contains the needed informations: which kernel image to run on which device, with which DT/rootfs, and what commands to run in userspace, etc...
- Runs KernelCI jobs: only boot.
- Provides an API to do many thing. Among them: sending jobs and querying results.

Buildroot

- `https://github.com/free-electrons/buildroot-ci`
- Used to make only rootfs.
- Is configured to provided the needed tools such as iperf, bonnie++, etc...
- Builds fresh rootfs ~~every night~~ for all the needed architectures: ARMv4, ARMv5, ARMv7, ARMv8.
- Stores them in a LAVA-accessible folder in the farm.
- Small overlay over vanilla buildroot:
  - `./build_rootfs.sh` to build the rootfs
  - `./configs/ci_*_defconfig` are the differents defconfigs.
  - `./board/ci/*` contains among other small stuff, the `busybox.config` file.

The custom test tool

- `https://github.com/free-electrons/custom_tests_tool`
- Has job templates and knows how to send them.
- Has a list of devices in the `boards.py` file containing the non-changing data
- Can fetch latest builds URLs from KernelCI through its API.
- Can upload custom artifacts useable for tests.
- Can ask LAVA to notify given emails addresses when the job ends.
- Needs an access to the LAVA API, an SSH access to the LAVA server to upload artifacts, and a KernelCI API token to go fetch the latest builds if no custom artifacts are provided.
- Keeps a `boards.py` file to store the boards configuration.

The test suite

- `https://github.com/free-electrons/test_suite`
- Is a bunch of shell scripts that will be run on the devices.
- Also needs YAML files to interface with LAVA.
  This file basically describes a list of commands to be run, and usually, it only calls the script.
- In practice, generic YAML files were made for the test suite, but there may be a need for specific ones in the futur.

Example:

```
metadata:
    format: Lava-Test Test Definition 1.0
    name: first-test
run:
    steps:
        - lava-test-case first-test --shell ./scripts/first_test.sh $DEVICE
```

Overview

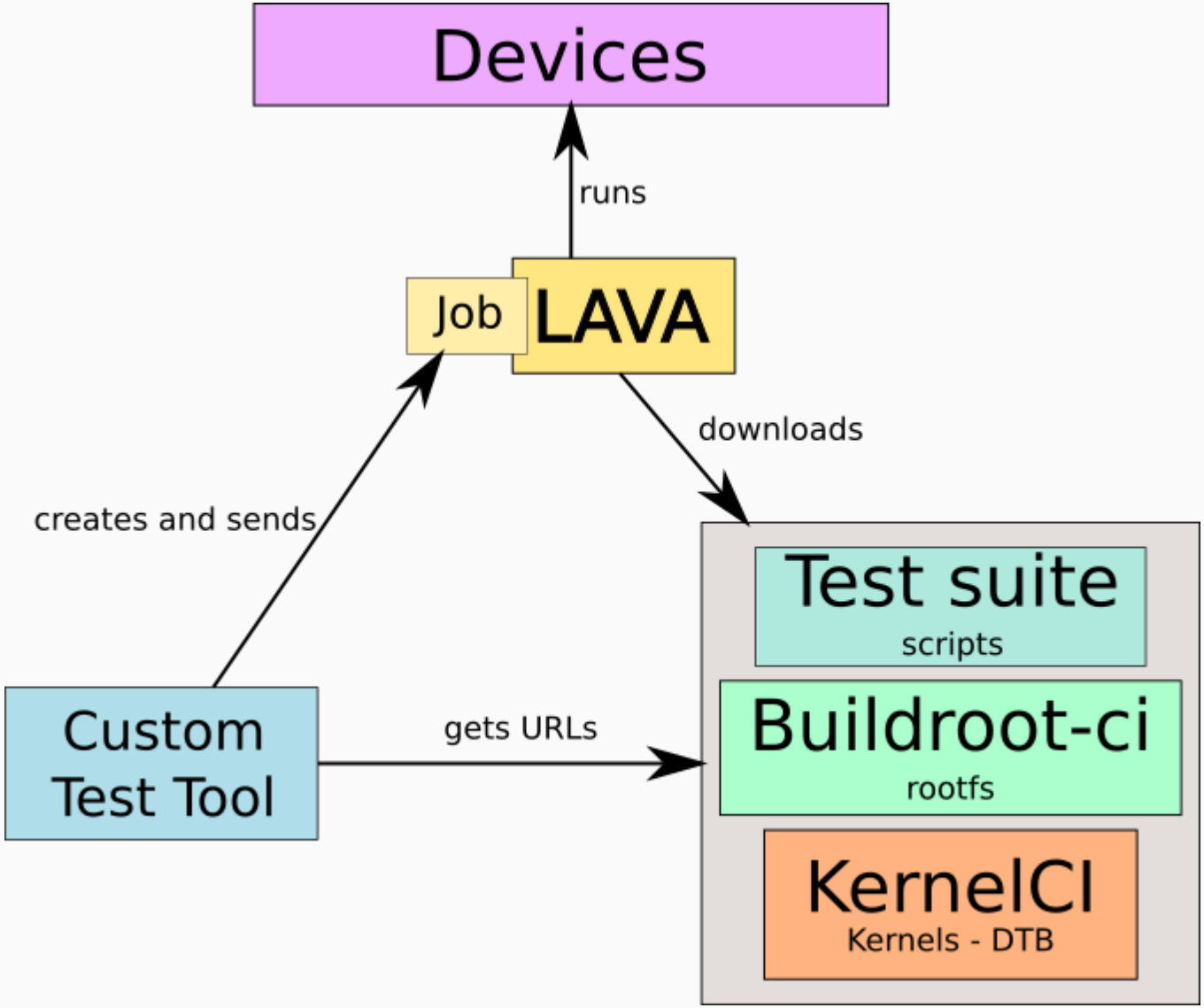

Figure 1: Architecture

Workflow

Daily jobs:

- A cron job runs at 23:23 to:
  - ~~Build fresh rootfs.~~
  - Launch custom tests on every boards with there default configuration.
- Another cron is run at 11:00 to get the results of the last 24 hours and send mails given the failed tests and which email address is subscribed to what board (in the `boards.py` file).

Workflow (2)

On demand jobs:

- `./ctt.py -b all`
  sends jobs to all boards with their default tests (it's what the daily cron job does)
- `./ctt.py -b beaglebone-black -m network`
  sends only the network test using the multinode job template (-m) to the beaglebone black
- `./ctt.py -b armada-7040-db armada-8040-db -t first_test --kernelci-tree next --no-send`
  stores the jobs (--no-send) for the 7040 and 8040 devices, to run only the first_test test on the latest KernelCI build of the next tree

How do tests work?

LAVA v2 jobs

- Jobs are YAML files that, combined with the device-types and devices dictionaries in the LAVA configuration, describe what to do with which device.
- It contains:
  - Which device(s) to use
  - The artifacts informations (URL, type, ...)
  - How to boot them (Uboot, fastboot, ramdisk, nfs, shell strings to expect, ...)
  - The tests to run once booted.
  - Some metadata and various job informations (priority, notifications, timeouts, ...)

How do LAVA runs test?

- Before bringing up the board, LAVA:
  - Fetches the artifacts.
  - Applies the needed modifications (append DTB, mkimage, modules, ...)
  - Also downloads the test suite and puts it in the rootfs. This moment also makes some magic to provide some helpers useable in the test scripts.
  - Finally connects itself and powers up the board.
- After the board is booted:
  - Runs each test described in the job definition.
  - For each call to the ‘lava-test-case‘ command, creates the corresponding results objects. It’s thus important to call it at least once, or you’ll have a job with no results, apart from the log.

How to write tests?

- Tests are in the `test_suite` repo.
- You basically just need to write your script in the `scripts` folder.
- Don't forget to give it the `.sh` extension, and for multinode jobs, they must follow this naming convention: `testname-laptop.sh` and `testname-board.sh`.
- Writing a single test is easy, but it's more complicated to write MultiNode jobs since you need to synchronize them.
- In MultiNode, you have access to helper commands [1]:
  - `lava-send <message-id> [key1=val1 [key2=val2] ...]`
  - `lava-wait <message-id>`

[1] Full reference:
`https://validation.linaro.org/static/docs/v2/multinodeapi.html`

Full MultiNode example

Server side:

```
#!/bin/sh
iperf -s &
echo $! > /tmp/iperf-server.pid
IP=‘ip route get 8.8.8.8 | head -n 1 | awk ’{print $NF}’‘
echo $IP
lava-send server-ready server-ip=$IP
lava-wait client-done
kill -9 ‘cat /tmp/iperf-server.pid‘
```

Client side:

```
#!/bin/sh
lava-wait server-ready
server=$(cat /tmp/lava_multi_node_cache.txt | cut -d = -f 2)
iperf -c $server
# ... do something with output ...
lava-send client-done
```

Collecting the results

- LAVA's notification system can send emails if asked. That's the way results are reported when sending on demand jobs.
- A daily cron job fetchs the results and aggregates them in an personalized email to each address in the `boards.py` file of the custom test tool.
- A dashboard is hourly refreshed presenting the devices and the tests in a table with colors and links to the jobs [2].
  The code lives in the custom test tool repo, under the `./dashboard` folder.

[2] It's running here: `http://farm:5000`

Sum up, improving, contributing...

The READMEs should already explain much of what you'll need.

- To add more tools to the rootfs:
  `https://github.com/free-electrons/buildroot-ci`
- To add more boards once they are in LAVA, or add more tests to a board, or to modify the job templates, or to improve the dashboard, or simply to subscribe to a device:
  `https://github.com/free-electrons/custom_tests_tool`
- To add more tests in the test suite:
  `https://github.com/free-electrons/test_suite`

Questions

Question?