Building a board farm: Continuous Integration and remote control

Quentin Schulz, Antoine Ténart
quentin.schulz@bootlin.com
antoine.tenart@bootlin.com
quentin.schulz@bootlin.com

Embedded Linux engineer at **Bootlin**

Former intern for creating a board farm for kernel continuous integration at **Bootlin**

- Embedded Linux specialists.
- Development, consulting and training (materials freely available under a Creative Commons license).
- [http://bootlin.com](http://bootlin.com)

Living in **Toulouse**, south west of France.
Antoine Ténart

- antoine.tenart@bootlin.com
- Embedded Linux engineer at Bootlin.
- Contributions
  - Kernel support for the Marvell Berlin ARM SoCs.
  - Kernel support for the Annapurna Labs ARM64 Alpine v2 platform.
- Living in **Toulouse**, south west of France.
Introduction

Quentin Schulz, Antoine Ténart
quentin.schulz@bootlin.com
antoine.tenart@bootlin.com
What is continuous integration?

Continuous Integration (CI) is a software engineering practice in which isolated changes are immediately tested and reported on when they are added to a larger code base. The goal of CI is to provide rapid feedback so that if a defect is introduced into the code base, it can be identified and corrected as soon as possible.

Three components: continuous builds, test automation and processing of the test results.

Source: TechTarget.com
Why does the kernel need it?

- Lots of different platforms (especially in the ARM world)
- Hard to test all the changes on all platforms
- Very frequent changes made by the community: new Linux release every two months, thousands of changes
- Need to detect regressions early
- Intel 0-day build bot is mainly for x86 platforms
Why do we need it?

- Bootlin contributes to ARM platforms upstream support
- Cooperation with several ARM processor vendors
- Many Bootlin engineers are maintainers of ARM and ARM64 platforms
  - Grégory Clement: Marvell EBU
  - Maxime Ripard: Allwinner
  - Alexandre Belloni: Atmel
  - Antoine Ténart: Annapurna Labs
- Keep track of modifications impacting the platforms we maintain
Components of continuous integration

Quentin Schulz, Antoine Ténart
quentin.schulz@bootlin.com
antoine.tenart@bootlin.com

© Copyright 2004-2018, Bootlin.
Creative Commons BY-SA 3.0 license.
Corrections, suggestions, contributions and translations are welcome!
Components of continuous integration

- Continuous build
- Test automation
- Processing
KernelCI

Quentin Schulz, Antoine Ténart
quentin.schulz@bootlin.com
antoine.tenart@bootlin.com
KernelCI - Overview

- Detects regressions before reaching users
- 2.000+ boot tests per day on 200+ unique boards

- Continuous build
  - Test automation
  - Test
  - Aggregate results

https://kernelci.org
Tracks ~20 kernel git repositories for changes

- torvalds/linux.git
- arm/arm-soc.git
- next/linux-next.git
- davem/net-next.git
- stable/linux-stable.git
- ...
KernelCI - Builder

- Builds kernels from tracked repositories
- Automatically builds all defconfigs for ARM, ARM64 and x86 (and their associated device trees, if any)
KernelCI - Connection with test automation

▶ Works with contributing labs
▶ Sends boot tests to labs, collects result and notifies maintainers of failures
Test automation - Software

Quentin Schulz, Antoine Ténart
quentin.schulz@bootlin.com
antoine.tenart@bootlin.com
Test automation

- Controls boards
- Launches tests on the boards
- Validates the tests and gathers the results
LAVA
KernelCI labs should use Linaro Automated Validation Architecture (LAVA) which:

- Controls boards
- Automates boot, bootloader, user-space, ... testing
- Runs tests simultaneously on all boards
- Provides API for full automation
- Validates tests
- https://wiki.linaro.org/LAVA
Organized in a master - dispatchers fashion

Only 1 master working with N dispatchers

The master controls the farm
  - It handles the API and receives the tests to run
  - It schedules the tests to run

A given dispatcher handles a set of boards
  - It has the boards' configuration files
  - It is physically connected to the boards and controls them
  - It runs the tests

We chose to host the master and our only dispatcher on the same machine
$ cat /etc/lava-dispatcher/devices/sun5i-r8-chip_01.conf

device_type = sun5i-r8-chip
hostname = sun5i-r8-chip_01
hard_reset_command = pduclient --daemon localhost --hostname drawer6
                           --command reboot --port 03 --delay 2
power_off_cmd = pduclient --daemon localhost --hostname drawer6
                           --command off --port 03 --delay 2
connection_command = telnet localhost 6063
$ cat /etc/lava-dispatcher/device-types/sun5i-r8-chip.conf
client_type = bootloader
send_char = False

z_load_addrs =
    0x42000000
    0x43300000
    0x43000000

boot_cmds_ramdisk =
    setenv autoload no,
    setenv kernel_addr_r '{KERNEL_ADDR}',
    setenv initrd_addr_r '{RAMDISK_ADDR}',
    setenv fdt_addr_r '{DTB_ADDR}',
    setenv ethact 'asx0',
    setenv loadkernel 'tftp ${kernel_addr_r} {KERNEL}',
    setenv loadinitrd 'tftp ${initrd_addr_r} {RAMDISK}; setenv initrd_size ${filesize}',
    setenv loadfdt 'tftp ${fdt_addr_r} {DTB}',
    setenv bootargs 'console=ttyS0,115200 earlyprintk root=/dev/ram0 ip=dhcp',
    setenv bootcmd 'usb start; dhcp; setenv serverip {SERVER_IP}; run loadkernel; run loadinitrd; run loadfdt; {BOOTX}',
    boot

bootloader_prompt = =>

boot_options =
    boot_cmds

[boot_cmds]
default = boot_cmds
Test automation - Hardware

Quentin Schulz, Antoine Ténart
quentin.schulz@bootlin.com
antoine.tenart@bootlin.com
Power supply control
Hardware - Power supply control

We need remotely controlled power supplies:

Figure: Power Distribution Unit

Figure: Remotely controlled relays

Figure: Network controlled multi-sockets
We chose remotely controlled relays because of:

- their cheap price
- the number of ports
- their small footprint
- their documented TCP protocol
- their support for virtually any power supply (you just need a wire)
Power supply
Three different types of boards:
- 5V powered boards
- 12V powered boards
- full ATX powered boards

We separate those in two kinds:
- non-ATX supplied boards
- ATX supplied boards
Different input voltages, two solutions:
  - one power supply per voltage with enough amperage
  - one power supply for all voltages with enough amperage

We chose ATX power supplies to get all voltages from one power supply
Hardware - Power supply of non-ATX supplied boards

![ATX Power Supply](image1)

**Figure:** ATX power supply

<table>
<thead>
<tr>
<th>AC Input Rating</th>
<th>DC Output Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC Input: 200V - 240V</td>
<td>DC Output</td>
</tr>
<tr>
<td>DC Output</td>
<td>+3.3V</td>
</tr>
<tr>
<td>Current: 5A</td>
<td>Max Load</td>
</tr>
<tr>
<td>14A</td>
<td>14A</td>
</tr>
<tr>
<td>Frequency: 47Hz - 63Hz</td>
<td>Maximum Combined Wattage</td>
</tr>
<tr>
<td>90W</td>
<td>300W</td>
</tr>
<tr>
<td>Total Power: 350W</td>
<td></td>
</tr>
</tbody>
</table>

**Figure:** ATX specifications

![TVS Diode](image2)

**Figure:** TVS diode
Hardware - ATX power supply specifics

▶ does not always supply power
▶ waits for a signal on #PS_ON or for #PS_ON to be put to the ground
▶ we need it to supply power all the time for non-ATX power supplied boards (the power from ATX power supply to the boards is controlled by per-board relays)
▶ we need to control when it supplies power to ATX power supplied boards

Figure: 24-pins ATX connector
Interaction with boards
Hardware - Connect to serial

- Mostly USB cable to board
- Lots of USB hubs
Hardware - Get and send files to boards

- TFTP protocol
- need of switches and Ethernet cables
Actual building of the board farm

Quentin Schulz, Antoine Ténart
quentin.schulz@bootlin.com
antoine.tenart@bootlin.com
Board farm - Specifications

- Specific location (200*100*75cm)
- Harmless to boards (material choices)
- Easy to use
- Allowing evolution
- As many boards as possible
Board farm
Board farm - Feedback

Quentin Schulz, Antoine Ténart
quentin.schulz@bootlin.com
antoine.tenart@bootlin.com

© Copyright 2004-2018, Bootlin.
Creative Commons BY-SA 3.0 license.
Corrections, suggestions, contributions and translations are welcome!
Launched on April 25th 2016
Currently 35 boards (estimated capacity of 50)
160k+ tests run
30+ unique devices support added to KernelCI
Board farm - Some challenges

- Many devices connected to the LAVA server which may have limitations. We had to recompile the kernel on this machine!
- All boards are different: specific U-Boot configuration, h/w modifications needed to automate the boot, very old bootloaders (U-Boot 1.1.1 from 2004)...
- Expect everything to fail: buggy serial connections, s/w services or machine configuration...
- LAVA assumptions may not match the hardware capabilities
Board farm - Documentation

- LAVA: https://validation.linaro.org/static/docs/index.html
- KernelCI: http://wiki.kernelci.org/
- Configure LAVA to receive tests from KernelCI:
  https://github.com/kernelci/lava-ci#configure-lava
- Adding a board to KernelCI:
  https://github.com/kernelci/lava-ci#add-board-to-kernelci
- Our articles on the matter: http://bootlin.com/blog/tag/lab/
Remote control

Quentin Schulz, Antoine Ténart
quentin.schulz@bootlin.com
antoine.tenart@bootlin.com
Why?

- Our farm knows how to handle boards, has a lot of them... but:
  - There is no direct access to the boards
  - Only tests sent to LAVA can perform actions on the boards
- Some boards owned only once
- Working remotely
LAVA Board Overseer

Quentin Schulz, Antoine Ténart
quentin.schulz@bootlin.com
antoine.tenart@bootlin.com
Lavabo

- Reuses the same tools LAVA uses
- Takes full control
- Authenticates users
- Interacts with LAVA
Lavabo reuses LAVA tools
Lavabo’s architecture

- client-server model
- server must be on the same machine as where LAVA server is hosted
- no support for multi-node LAVA
- one dedicated SSH user on the server
- one SSH key per lavabo real user
- LAVA’s `connection_command` for all devices must be telnet
- no support for rootfs on NFS
## Typical workflow

```plaintext
$ lavabo list
status     job         offline_since               hostname          offline_by
---------- ----- ------------------------ -------------------- ------------
idle        idle alpine-db_01
offline     Tue Oct 4 14:23:51 2016 alpine-v2-evp_01
idle        armada-370-db_01
idle        Wed Sep 21 15:46:56 2016 armada-3720-db_01
idle        sun5i-r8-chip_01
idle        sun8i-a33-sinlinx-sina33_01
idle        sun8i-a83t-allwinner-h8homlet-v2_01
```

$ lavabo reserve sun5i-r8-chip_01
$ lavabo upload mykernel sun5i-r8-chip.dtb myrootfs
File(s) successfully sent to lavabo-server.
$ lavabo reset sun5i-r8-chip_01
```
$ lavabo serial sun5i-r8-chip_01
Try 1 to connect to serial failed. 4 attempts remaining.
You have now access to the serial of sun5i-r8-chip_01.
Escape character is '^[']'.
U-Boot SPL 2016.01-g67a66a1.dirty (Mar 09 2016 - 12:04:29)
DRAM: 512 MiB
CPU: 1008000000Hz, AXI/AHB/APB: 3/2/2
Trying to boot from NAND

U-Boot 2016.01-g67a66a1.dirty (Mar 09 2016 - 12:04:29 +0100) Allwinner Technology
CPU: Allwinner A13 (SUN5I)
I2C: ready
DRAM: 512 MiB
NAND: 8192 MiB
video-mode 720x480-24@60 not available, falling back to 1024x768-24@60
Setting up a 720x480i composite-ntsc console (overscan 40x20)
[...] Hit any key to stop autoboot: 0
=>

$ lavabo power-off sun5i-r8-chip_01
$ lavabo release sun5i-r8-chip_01
Lavabo - Conclusion

- Some limitations
- GNU GPLv2 licensed
- https://github.com/bootlin/lavabo
- Let’s play!
Thanks! Questions?

Slides under CC-BY-SA 3.0