Linaro Connect
MADRID 2024 | MAY 14-17 2024

MAD24-206 Boot time optimization project

Daniel Lezcano <daniel.lezcano@linaro.org>

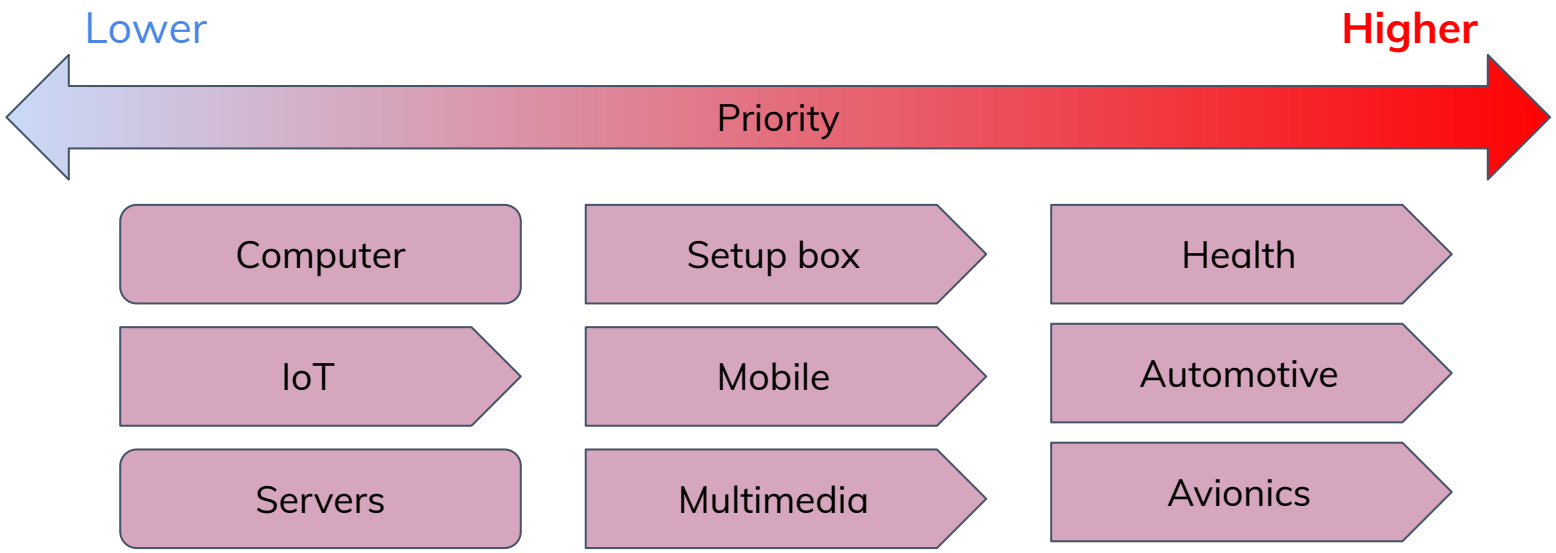
Boot time

Problems
○ Some systems must be ready ASAP after being powered up ○ Open source components can introduce boot up latencies and fixes must be investigated and carried on versions ○ Knowledge is lost with project switches or turn over ○ No interactions between firmware / kernel / userspace for the boot process

Boot time

Goals
○ Optimize the boot time duration in firmware, kernel and userspace ○ Track down boot time regression automatically ○ Build a knowledge base to accelerate new platforms boot time optimizations ○ Long term support
Proposal
○ Choose a reference platform for boot time optimizations (must be representative) ○ Code Linaro for automated non regression testing and diagnostic (AI?) ○ Code Linaro for knowledge base ○ Upstream changes for long term support

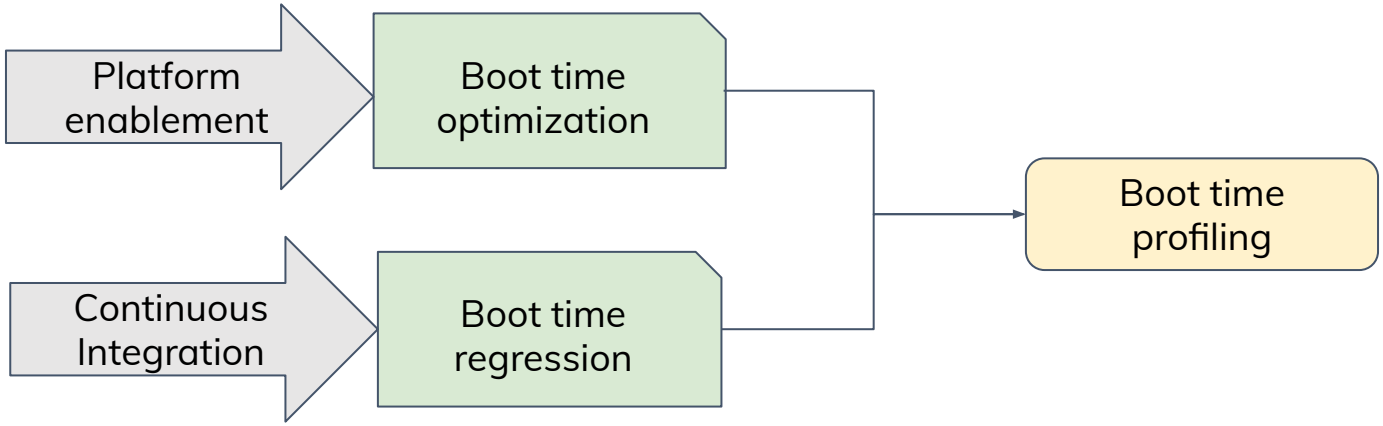
Boot time priority trend



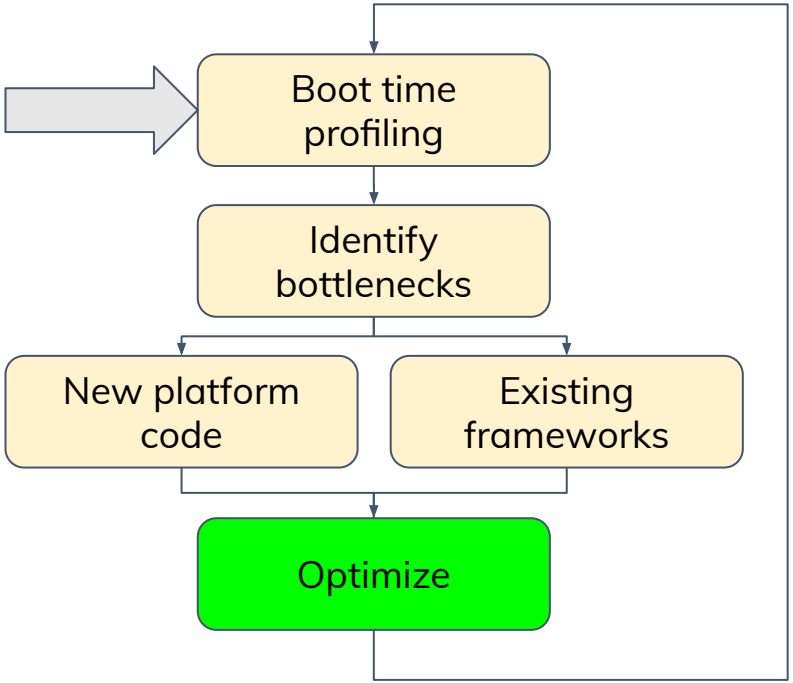
: Tends to higher priority

: Priority stable

Boot time target is a continuous process

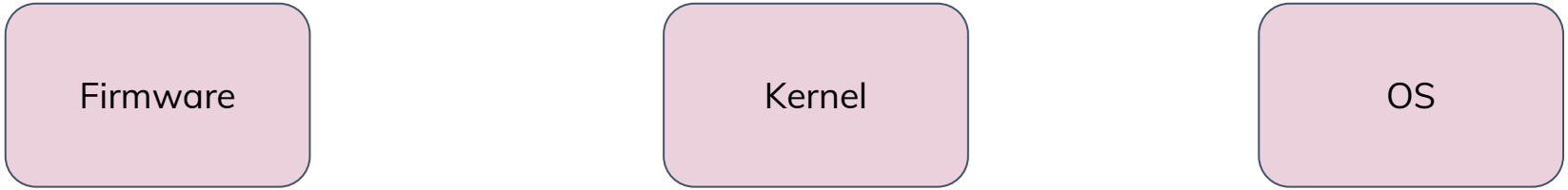


Boot time optimization process is complex

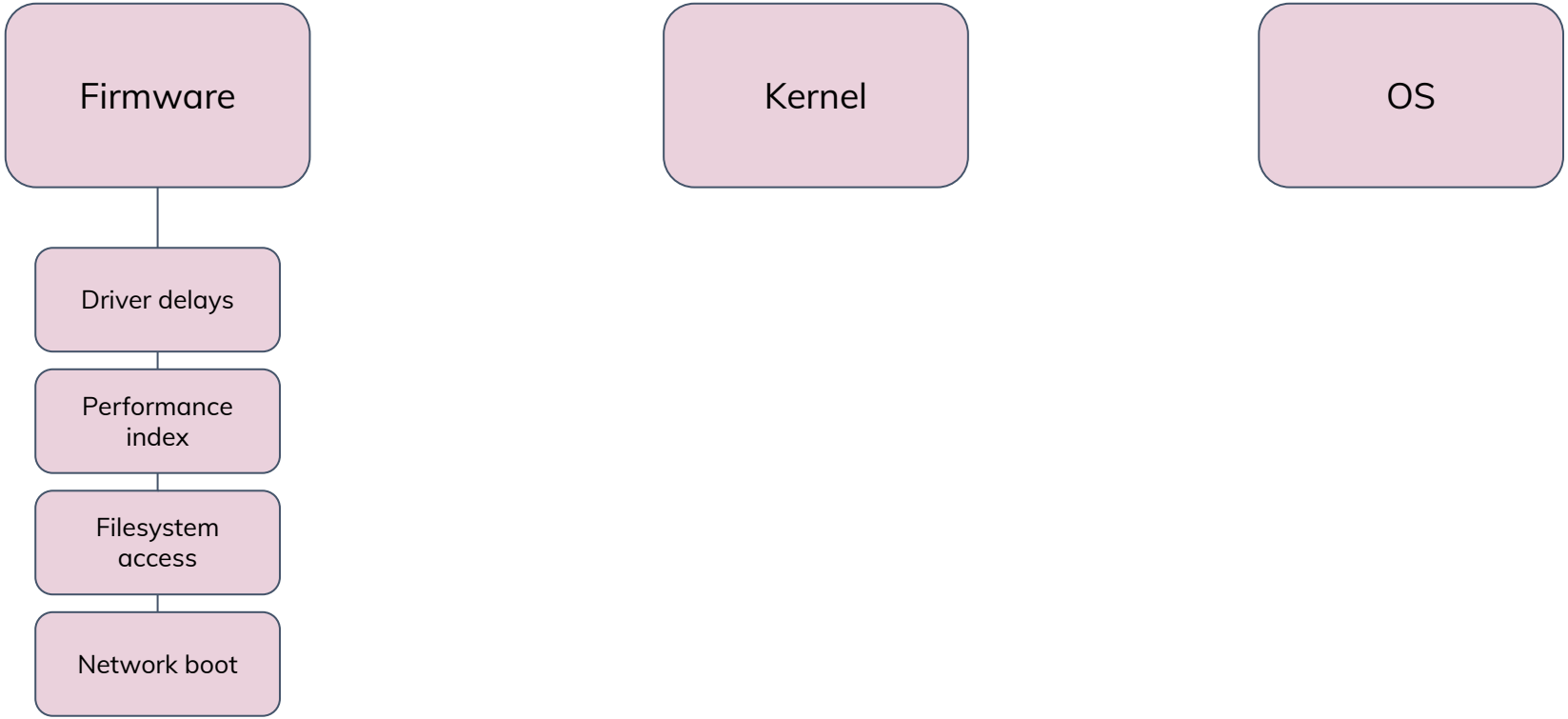


- High level of expertise
  - understanding of all low level components
- Full stack understanding
  - firmware + kernel + userspace
- Recognized open source actor
  - Optimizations must be upstream to prevent repeating the operation again and again
- Capitalization of the knowledge
  - Document the optimizations for next platforms

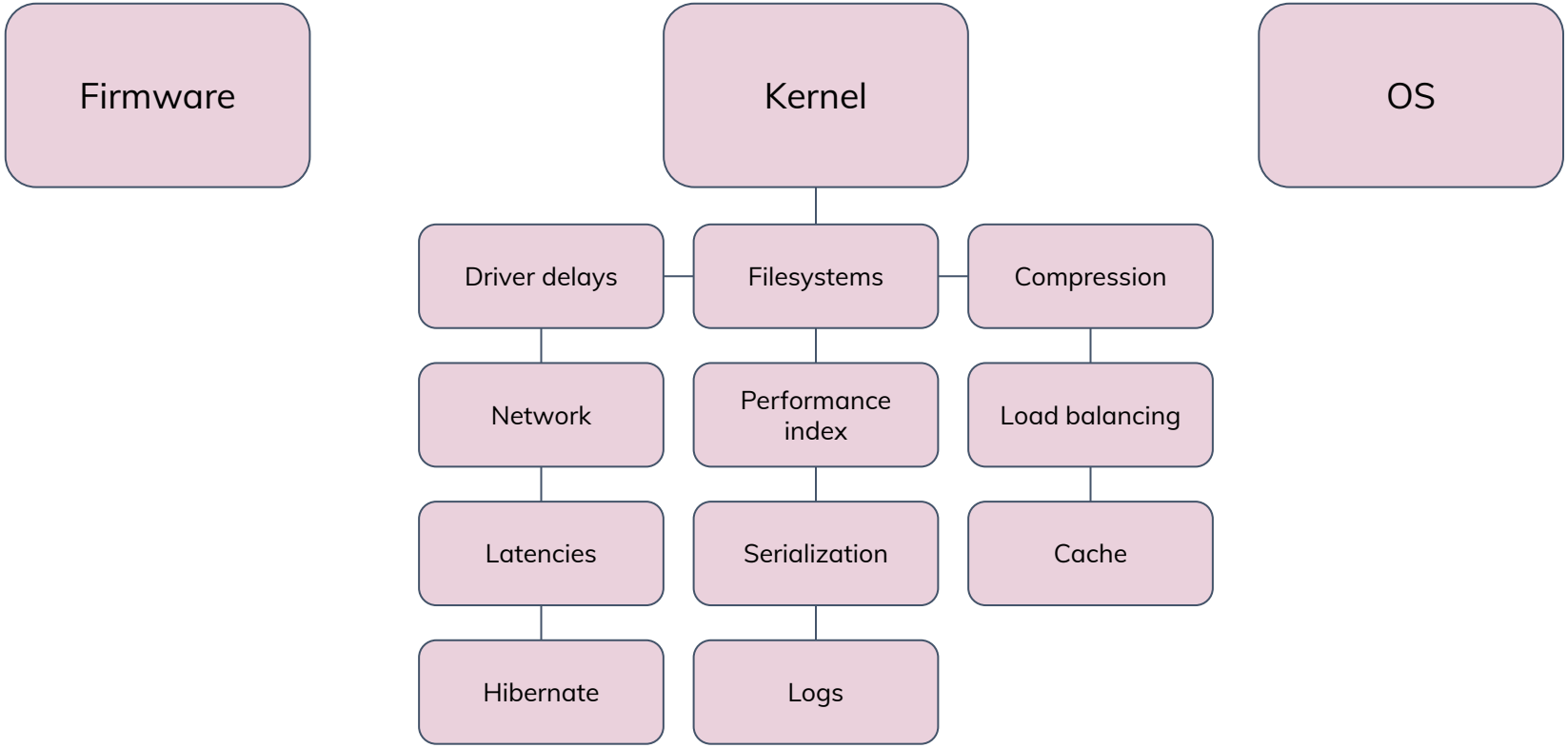
Boot time



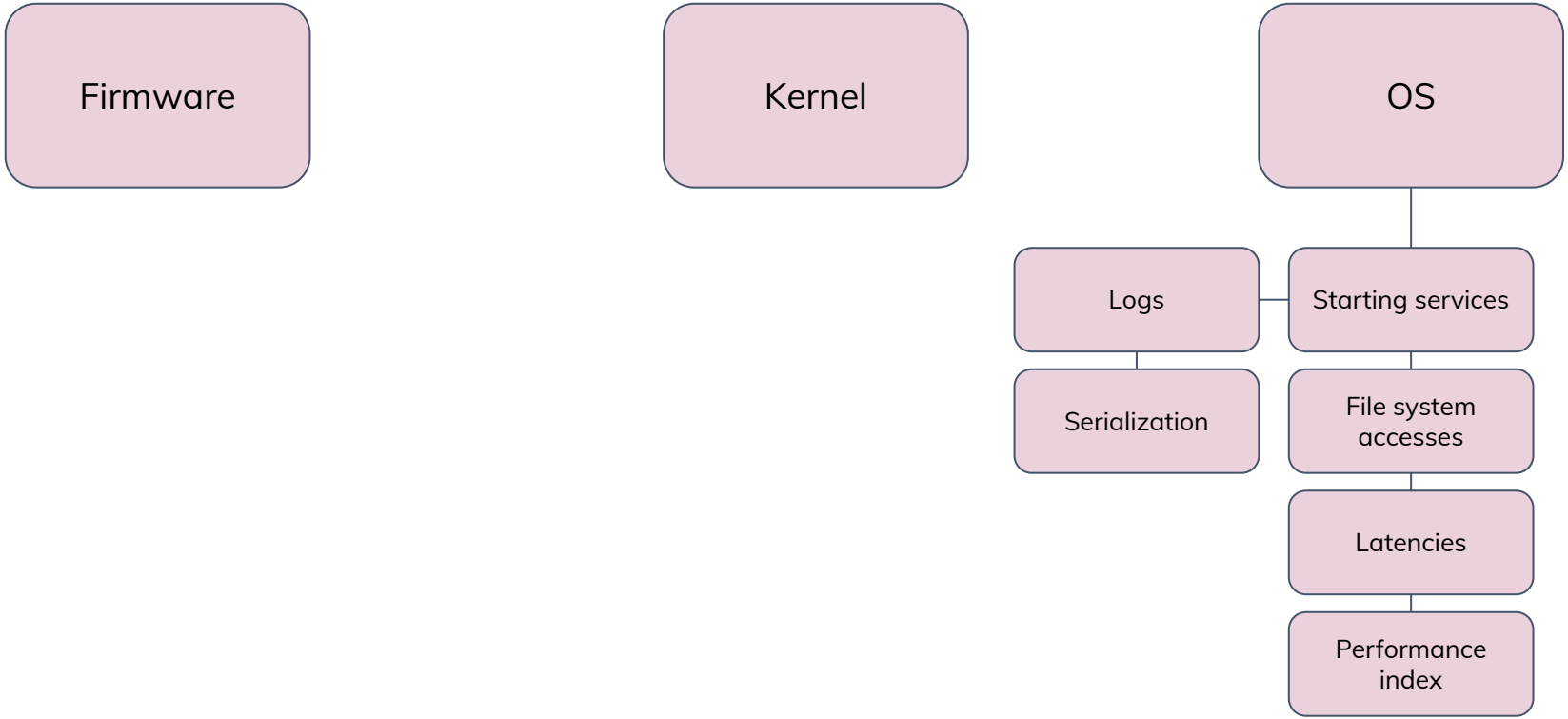
Boot time


Firmware
Kernel
OS
Driver delays
Performance index
Filesystem access
Network boot

Boot time


Firmware
Kernel
OS
Driver delays
Filesystems
Compression
Network
Performance index
Load balancing
Latencies
Serialization
Cache
Hibernate
Logs

Boot time


Firmware
Kernel
OS
Logs
Starting services
Serialization
File system accesses
Latencies
Performance index

Boot time : Continuous Integration

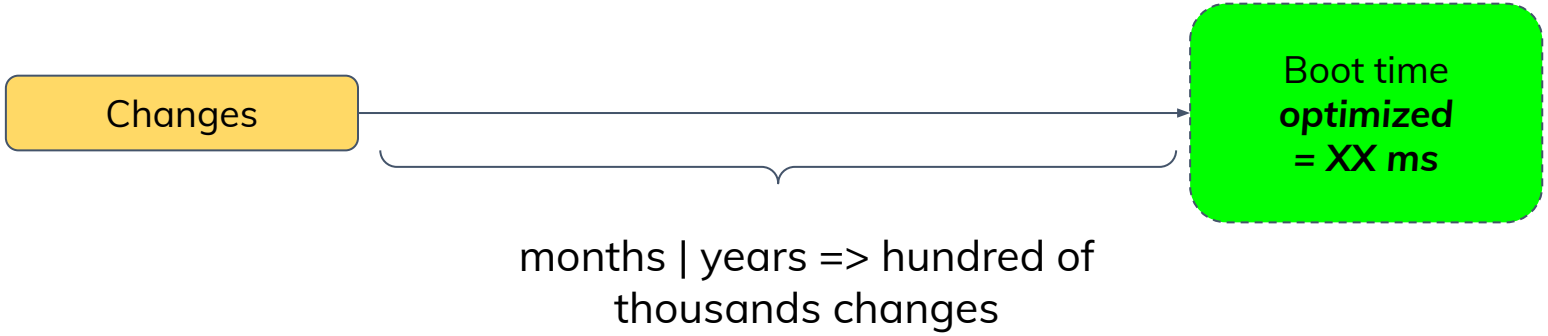
- A boot time optimization is a moving target

Changes

Boot time
optimized
= XX ms

Boot time : Continuous Integration

- A boot time optimization is a moving target



Boot time : Continuous Integration

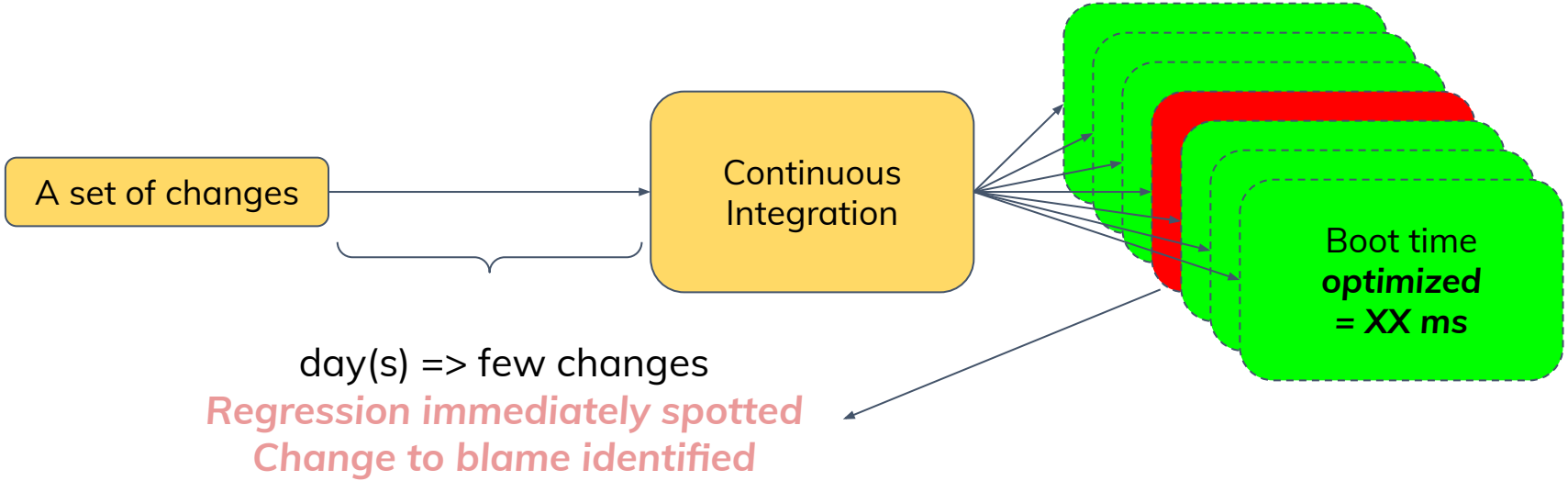
- A boot time optimization is a moving target

months | years => hundred of thousands changes

Which ones introduced the regressions?

Boot time : Continuous Integration

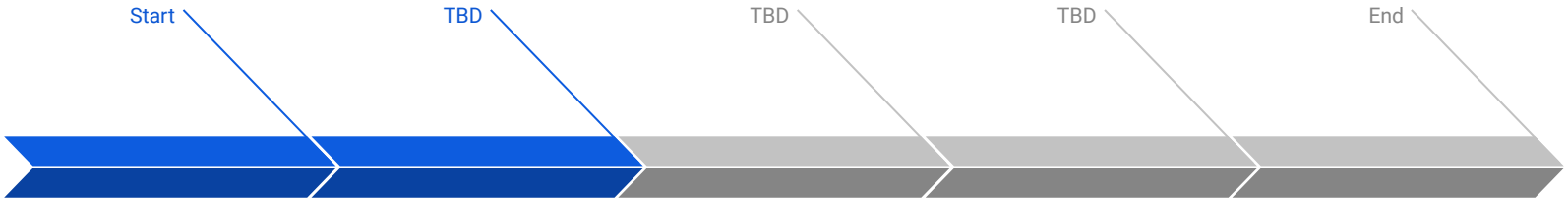
- A boot time optimization is a moving target



Boot time : Existing tools

- Firmware
  - UEFI to pass boot time information to the kernel
- Kernel level
  - CONFIG_BOOTTIME_TRACING
  - CONFIG_DEBUG_LL
  - CONFIG_EARLY_PRINTK
  - grabserial
- Userspace
  - Linux : bootchart2, systemd-analyze
  - Android : bootchart

Boot time : Project roadmap



Frame definition

A phase of definition of the boot time objectives, the platforms for validation, the layers of the stack and the targets. Creation of the knowledge base. Define the baselines.

Continuous Integration

Automated integration trees, job submissions, scripts, boot time graphics and automatic regression detection

Firmware **Kernel** **Operating System**

Analyze and spot the bottlenecks. Get measurements. Optimize and upstream the changes.

Document and store in the knowledge base.

Boot time : Project first steps

As a starting point, stay focused on kernel and lower layers

- Generic way to measure boot time and pass information to the kernel
  - Firmware measures the devices + bootloader loading + execution timings
  - Bootloader measures kernel image loading + decompression + execution timings
  - Kernel measures subsystems timings
  - Kernel exports all data to userspace via a RO file system (eg. debugfs)

  ⇨ No information on tty

- Continuous boot time measurement and regression analysis
  - CI Loop : Code Linaro + LAVA + SQL DB + Grafana (until SQUAD fully functional)
    - Infrastructure available
  - Rely on boot time kernel formatted measurements
    - No extra tools for this part

Boot time : Potential areas of improvement

Firmware	
Boot chain timing	Identify per firmware timings if there are several
Device initialization	Let know the kernel what devices are already initialized
Boot information	Standardize boot time information to collect with userspace tools

Boot time : Potential areas of improvement

Kernel	
Latency	Keep the boot CPU to the highest performance state without going idle during the boot phase which includes the userspace telling the kernel the boot phase ended which in turn remove the latency constraint
Logs	Remove pointless logs
File system	Investigate if BFQ is enabled after all CPUs are up
Parallelization	Investigate async probes (PROBE_PREFER_ASYNCHRONOUS)
Driver delays	Investigate busy loops in drivers initialization
Compression	Use LZ4 and reduce initramfs size
Parallelization	Start thermal zones monitoring in parallel
Memory Initialization	Big memory system takes time to initialize

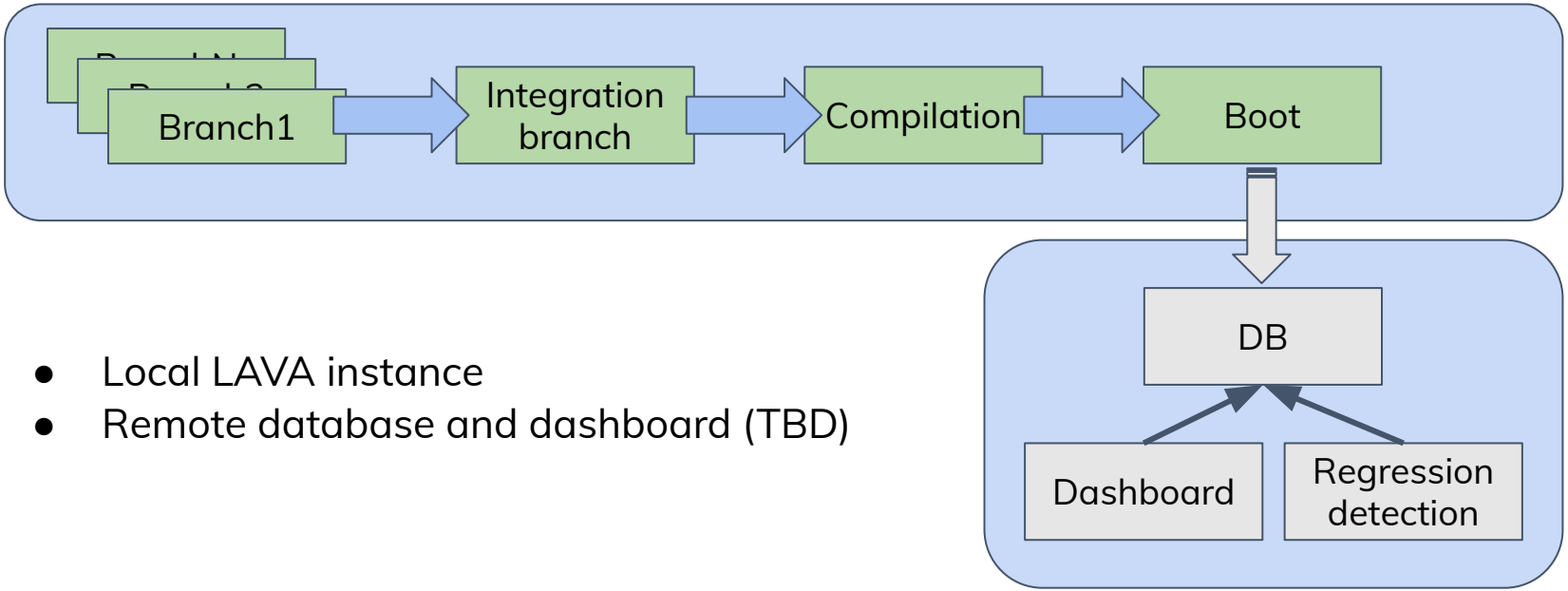
Boot time : Potential areas of improvement

Operating system	
File system access	Investigate how readahead is used to read services at boot time
Logs	Remove pointless logs
Services	Daemons started on request
Parallelization	Parallel kernel modules loading (modprobe changes)
Parallelization	Services started in parallel (dependency checks)

Boot time optimizations plan

- Part 1 : CI Loop
  - Implementation of missing firmware boot information (UEFI API in u-boot) => boot time measurement unification
  - Automatic detection of branches changes, including upstream => Integration branch + compilation + boot
  - Boot time measurements stored in a database => history
  - Graphic rendering with time series figure => dashboard
  - Automatic detection of boot time regression => email
  - [Identification of commit id => email] (nice to have feature but complex)

Boot time optimization - Part I



- Local LAVA instance
- Remote database and dashboard (TBD)

Boot time optimizations plan

- Part 2 : Engineering and optimization
  - Probe time optimizations (firmware + kernel collaboration)
  - modules loaded in parallel (userspace + kernel)
  - Fix mdelay in device initialization routines (firmware + kernel)
  - Daemons started on request (userspace)
  - CPU latencies optimization (kernel)
  - Memory hotplug for big memory configuration
  - etc ...

This second part has to rely on the first part in order to track down the evolution of the boot time during the optimization process. Also sticking to the upstream kernel helps to detect when outside contributions improved or degraded the boot timings.

Boot time optimizations plan

- Proof of concept
  - https://linaro.atlassian.net/browse/STG-5941
- Plan of Record after Connect
  - Connect is the place to discuss about it
- Interested ?
  - Let us know
  - Propose your board
  - Request your boot time optimization

Linaro Connect
MADRID 2024 | MAY 12-17 2024

Thank you