Unifying GitLab CI approaches for the Linux Kernel

Helen Koike
Sebastian Fricke

LPC 2024

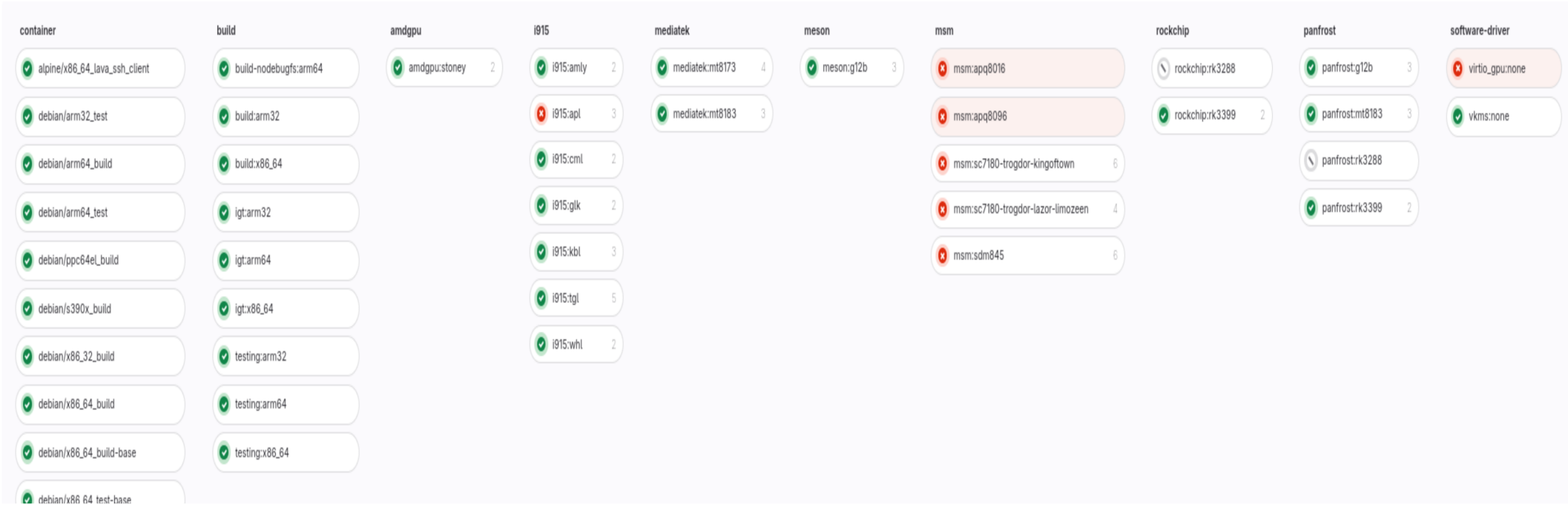
COLLABORA

What is DRM-CI?

DRM-CI: what is it?

- GitLab-CI pipeline to test DRM
- In kernel tree
- Uses Freedesktop infrastructure
- Annotated expectation files
- How to use:
  - Add your repository to gitlab.freedesktop.org
  - Request access to drm-ok group
  - Trigger pipelines

DRM-CI: current pipeline

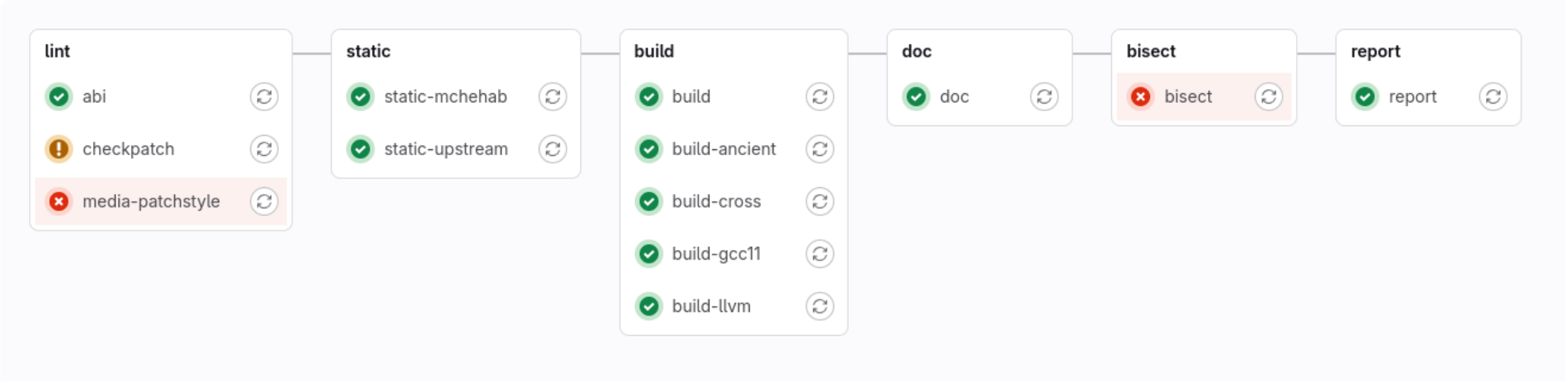


COLLABORA

How others are doing it?

Media-CI

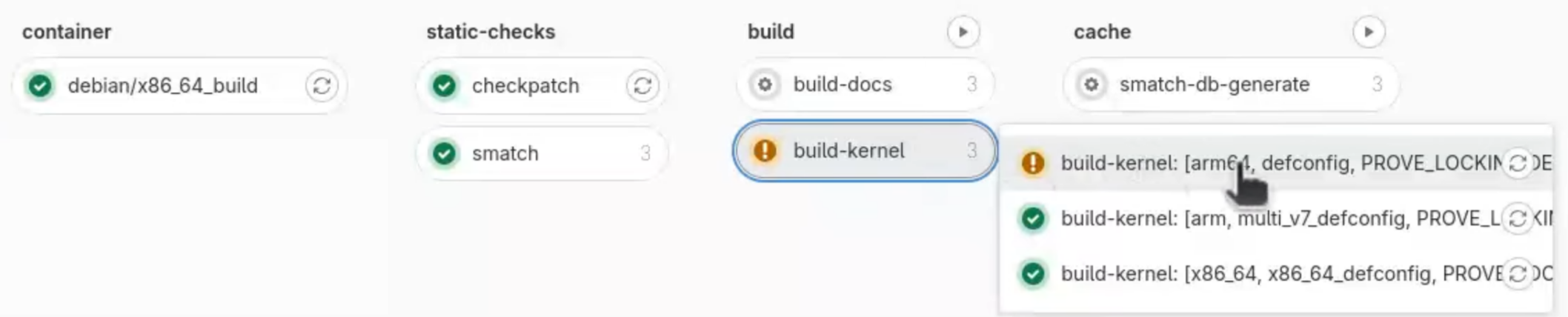
- Recent out-of-tree pipeline on Freedesktop



KCI-GITLAB: common base

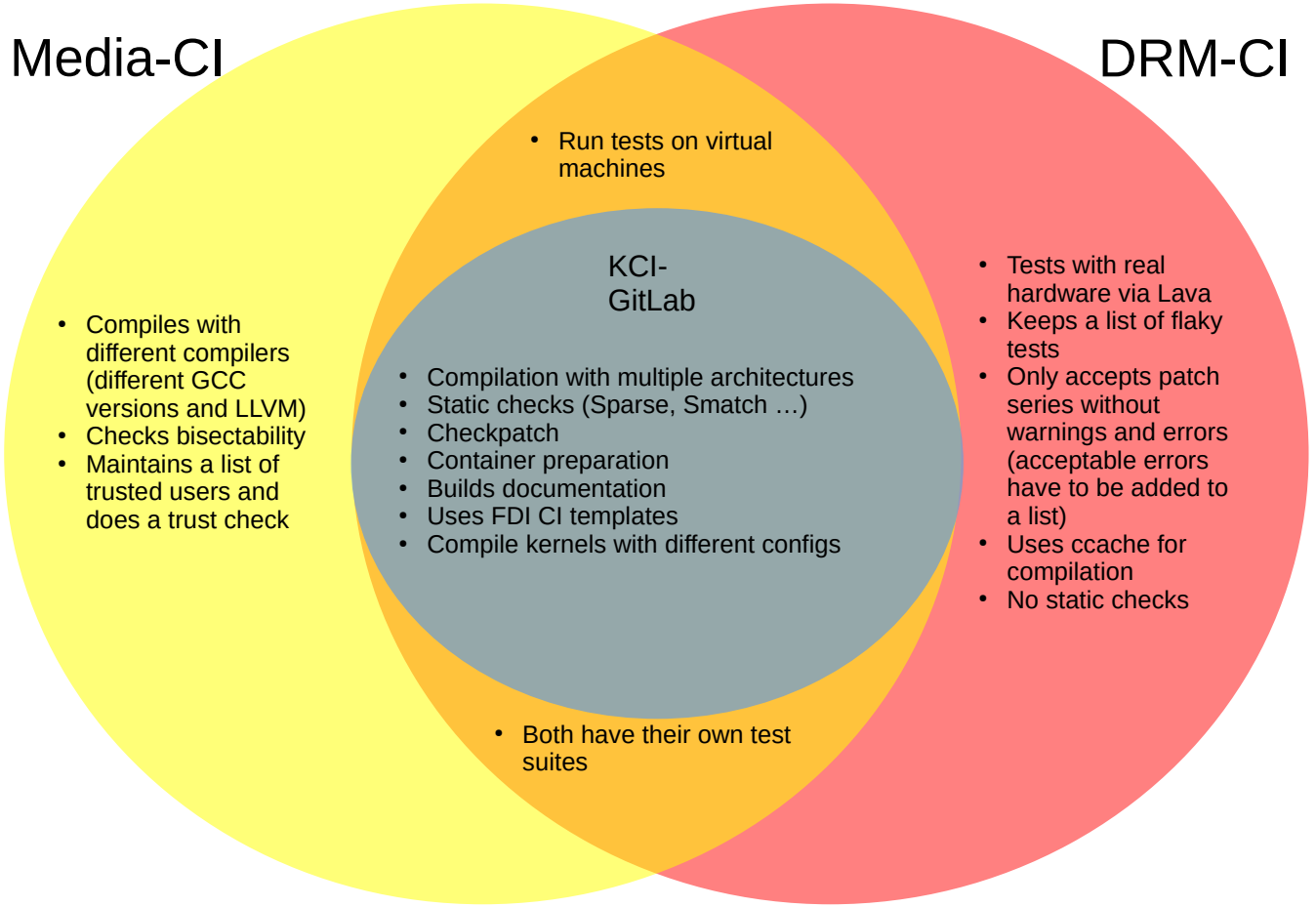
KCI-GITLAB: details

- Patch v1 submitted on 28th Feb 2024
- Meant to be common base
- Test-scenarios feature
  - Allow override jobs (in-tree)
  - Easily share test-scenarios



COLLABORA

Overview of differences and similarities


Media-CI
DRM-CI
Run tests on virtual machines
KCI-GitLab
Compiles with different compilers (different GCC versions and LLVM)
Checks bisectability
Maintains a list of trusted users and does a trust check
Compilation with multiple architectures
Static checks (Sparse, Smatch …)
Checkpatch
Container preparation
Builds documentation
Uses FDI CI templates
Compile kernels with different configs
Tests with real hardware via Lava
Keeps a list of flaky tests
Only accepts patch series without warnings and errors (acceptable errors have to be added to a list)
Uses ccache for compilation
No static checks
Both have their own test suites

How DRM-CI profit from KCI-GITLAB?

- Get static checks
- Easier to specify Test Scenarios (per driver, per maintainer, etc)
- Independent of platform
- Base for other tests (kselftests, kunit, etc...)

COLLABORA

What is the problem?

What is the problem?

- Duplicated and inefficient efforts
- Not being up to date with latest best practices
- Not benefiting or contributing to other’s work
- Individual infrastructure
- High entrance barrier to subsystems without an existing CI system

Why should you care?

- Reduced maintenance cost
- More reviewers to DRM-CI
- Access to other tests and infrastructure
- Contribute to the Linux testing ecosystem as a whole

Potential down sides

- Slower to make changes:
  - More reviews and tests required
  - Designing changes is more complicated
- Specific to GitLab

Steps to migrate to KCI-GITLAB

1) Step 1: Finish **upstreaming** KCI-GITLAB
    1) Move KCI-GITLAB to tools/ci/ folder with more generic script (“library” like)
    2) **Utilize these scripts from tools/ci in DRM-CI**
    3) Submit v2 of KCI-GITLAB
    4) Work estimation in total: 2 weeks FTE (2 days for 1.2)
2) Step 2: Move DRM-CI definition as a **Scenario to KCI-GITLAB**
    1) Dependent on KernelCI Labs to dispatch tests to devices
    2) **Re-implement dependencies from MesaCI in KCI-GITLAB**
    3) Write DRM Scenario in KCI-GITLAB
    4) Work estimation in total: 2 weeks FTE

COLLABORA

Questions for the audience

- What do you think about handling board lab access via the KernelCI? (instead of doing that via Mesa-CI)
- What are you currently missing in the DRM-CI?
- Do you have an idea for funding?
- How should we organize CI development and maintainance workflow?

Thank you!