

Addressing Mesa CI pain points (2nd edition)

What we've done, and where we are going

Eric Engestrom, Martin Roukala, Sergi Blanch Torné, Valentine Burley

Valve, Collabora



2026-09-28

Outline

Guiding principles

Practical challenges

What we've done

What now?

Annexes

What are we striving for?

Never merge regressions

- Merge action tied to CI results
- Put the cost of integration on the person making the changes

What are we striving for?

Never merge regressions

- Merge action tied to CI results
- Put the cost of integration on the person making the changes

Minimize the impact on the developer's workflow

- Short execution time
- No false positives nor negatives
- Good interface for starting desired tests and getting results

Outline

Guiding principles

Practical challenges

What we've done

What now?

Annexes

How to integrate with the developers' workflow?

Use the tools provided by the project's forge: **GitLab CI**

How to integrate with the developers' workflow?

Use the tools provided by the project's forge: **GitLab CI**

GitLab

- Web UI
- Merge-request-oriented contributions
- Contributors' roles are managed via the UI, repo & CI access control

How to integrate with the developers' workflow?

Use the tools provided by the project's forge: **GitLab CI**

GitLab

- Web UI
- Merge-request-oriented contributions
- Contributors' roles are managed via the UI, repo & CI access control

GitLab CI

- CI pipelines are graphs of jobs
- Test environments are built as containers
- Runners request jobs to execute from GitLab
- Pipelines can be run on git **push** (not per commit), or on a **schedule**

How to integrate with the developers' workflow?

“Every change must be tested”

- MRs are serialized: rebase, test, merge, pick the next MR
- GitLab didn't support this, and to annotate commits:
 - → Marge-bot script

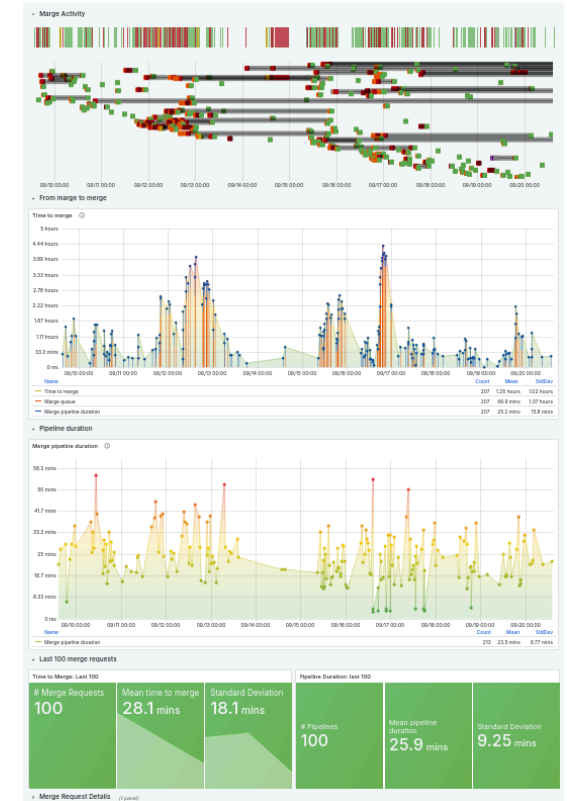
How to integrate with the developers' workflow?

“Every change must be tested”

- MRs are serialized: rebase, test, merge, pick the next MR
- GitLab didn't support this, and to annotate commits:
 - Marge-bot script

Problem: Serialization doesn't scale

- Marge pipeline 1h timeout (worst case, 24 MRs/day)
- Encourages creating big MRs rather than tons of small ones



Dashboard [From Marge to merge](#)

How to trigger a job (and the right one)?

History:

- “Always run everything” → too much resource usage, preventing merges

How to trigger a job (and the right one)?

History:

- “Always run everything” → too much resource usage, preventing merges
- “Run jobs of affected drivers” (file lists)

How to trigger a job (and the right one)?

History:

- “Always run everything” → too much resource usage, preventing merges
- “Run jobs of affected drivers” (file lists)
- “Run jobs manually (except Marge)”, but badly supported by GitLab UI

How to trigger a job (and the right one)?

History:

- “Always run everything” → too much resource usage, preventing merges
- “Run jobs of affected drivers” (file lists)
- “Run jobs manually (except Marge)”, but badly supported by GitLab UI
- `ci_run_n_monitor.sh` script added to the repo

How to trigger a job (and the right one)?

History:

- “Always run everything” → too much resource usage, preventing merges
- “Run jobs of affected drivers” (file lists)
- “Run jobs manually (except Marge)”, but badly supported by GitLab UI
- `ci_run_n_monitor.sh` script added to the repo

Problems:

- Not integrated in the GitLab web UI / not discoverable
- Project-specific, not standardized across freedesktop.org 🚧

How to only report on the changes of the MR?

Goal:

- CI results should present all **regressions** and **fixes**
 - Filter out **existing issues** found in the merge base

How to only report on the changes of the MR?

Goal:

- CI results should present all **regressions** and **fixes**
→ Filter out **existing issues** found in the merge base

How:

- Known issues are documented in tree using **text lists**
- Test results are compared to these known issues
- Unexpected test results are automatically retried **once**, and marked as **fail** / **flakes**
- Jobs are failed if they have “stable” results not matching expectations
- Failed jobs are retried **once** in **merge** pipelines

How to only report on the changes of the MR?

Problems:

- **Test flakiness allows merging regressions:**
 - Undocumented flakes cause issues in future MRs



Dashboard MesaCI false positives

How to only report on the changes of the MR?

Problems:

- Test flakiness allows merging regressions
 - Undocumented flakes cause issues in future MRs
- **Updating expectations is required from developers, with varying practices:**
 - A script is being written to automate it (ci-collate)
 - Will still require expert review and decision



Dashboard MesaCI false positives

How to dig into CI results?

Available information to humans, through GitLab's web UI:

- Overall acceptance result available as pipeline & job status (pass/fail)
- Job log contains details (boot, execution, ...)
- Job artifacts:
 - Machine-readable results (CSV)
 - Summary HTML pages (piglit jobs)

How to dig into CI results?

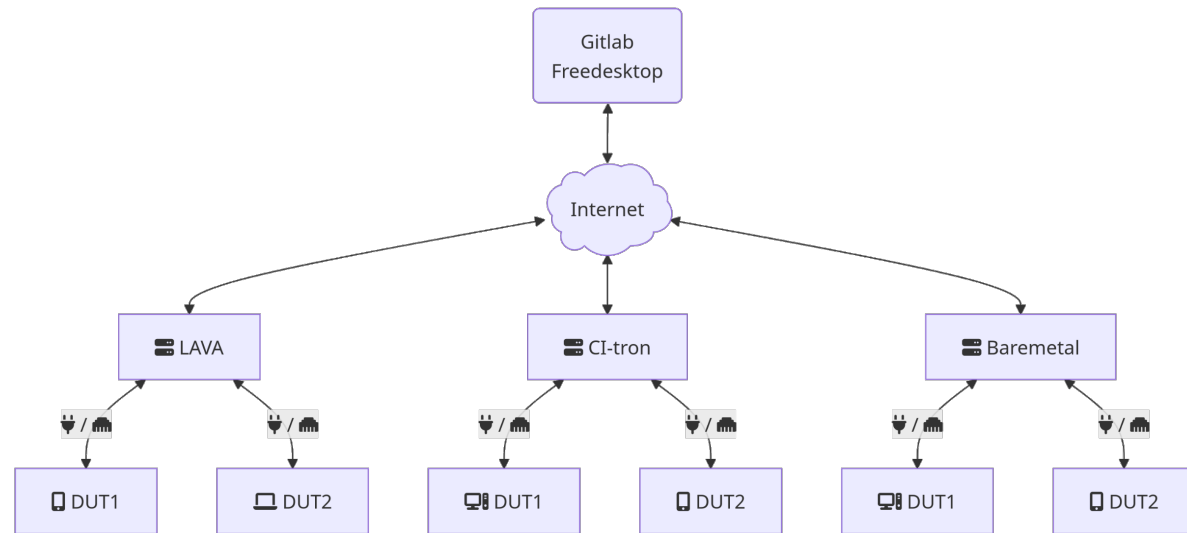
Available information to humans, through GitLab's web UI:

- Overall acceptance result available as pipeline & job status (pass/fail)
- Job log contains details (boot, execution, ...)
- Job artifacts:
 - Machine-readable results (CSV)
 - Summary HTML pages (piglit jobs)

Problem:

- Artifacts are not easily discoverable, unless the job log links directly to them
- Correlating job logs is very hard and slow, especially when browsing the web UI

How to expose test machines to GitLab?



3 infrastructures, but many farms available: Baremetal, LAVA, CI-tron

How to keep execution time short?

- **Run tests in parallel within a job (deqp-runner):**
 - Hang detection is difficult & unreliable
 - Not all test suites can be run in parallel / supported by deqp-runner

How to keep execution time short?

- Run tests in parallel within a job (deqp - runner)
- **Fractional job:**
 - Tests not run may regress
 - Mitigated by nightly full jobs, but maintenance cost
 - → Not all fractional jobs have a corresponding full job

How to keep execution time short?

- Run tests in parallel within a job (deqp - runner)
- Fractional job
- **Parallel over multiple machines:**
 - Multiplies the number of machines needed
 - Boot/setup overhead is multiplied by the number of machines

How to keep execution time short?

- Run tests in parallel within a job (deqp - runner)
- Fractional job
- Parallel over multiple machines
- **Skips lists of long tests:**
 - These tests can regress silently
 - Manual work to identify, add & remove tests from that list

How to keep execution time short?

- Run tests in parallel within a job (deqp - runner)
- Fractional job
- Parallel over multiple machines
- Skips lists of long tests

Dashboards and automated alerts

- Example: [DUT time per GitLab job](#)

How to share test machines?

Problem:

- Test machines (DUTs) are **shared** between:
 - Projects and users of gl.fد.o
 - Other GitLab instances, Kernel CI, GitHub

How to share test machines?

Problem:

- Test machines (DUTs) are **shared** between:
 - Projects and users of gl.fد.o
 - Other GitLab instances, Kernel CI, GitHub

Solutions:

- **Be kind:**
 - Limit usage by **running jobs you need**
 - Consider **delaying** stress test to USA nights / weekends
 - Check the **Merge queue**: with the filter **assignee=marge-bot**, or running `bin/ci/marge_queue.sh` script (spoiler)

How to share test machines?

Problem:

- Test machines (DUTs) are **shared** between:
 - Projects and users of gl.f.d.o
 - Other GitLab instances, Kernel CI, GitHub

Solutions:

- Be kind
- **Job prioritisation**
 - Generic runners: 
 - CI-tron:  
 - LAVA:  (good-enough)
 - Baremetal:  (deprecated, not planned)

How to share test machines?

Problem:

- Test machines (DUTs) are **shared** between:
 - Projects and users of gl.fdo
 - Other GitLab instances, Kernel CI, GitHub

Solutions:

- Be kind
- Job prioritisation
- **Preemption** (pausing lower-priority jobs)
 - Generic runners: Not applicable
 - HW runners: planned for 2026/2027 for CI-tron farms

How to give all the relevant information and not flood the reader with unnecessary details that should not be on the screen?



Best practices:

- Less-is-more: Collapse sections to hide usually-not-relevant information
- Use colors to **highlight** important events
- Print a summary at the end of the job log
- Push less-relevant information to artifacts

How to give all the relevant information and not flood the reader with unnecessary details that should not be on the screen?



Best practices:

- Less-is-more: Collapse sections to hide usually-not-relevant information
- Use colors to **highlight** important events
- Print a summary at the end of the job log
- Push less-relevant information to artifacts

Problems:

- We are limited by what GitLab allows (more on that later)

Outline

Guiding principles

Practical challenges

What we've done

What now?

Annexes

Roadmap: Recap from xdc2025

By XDC 2026:

- Job prioritisation (CI-tron & LAVA)

Roadmap: Recap from xdc2025

By XDC 2026:

- Job prioritisation (CI-tron **done**  & LAVA **deemed good-enough**)

Roadmap: Recap from xdc2025

By XDC 2026:

- Job prioritisation (CI-tron **done**  & LAVA **deemed good-enough**)
- Preemption support (CI-tron)

Roadmap: Recap from xdc2025

By XDC 2026:

- Job prioritisation (CI-tron **done**  & LAVA **deemed good-enough**)
- Preemption support (CI-tron) **not done** 

Roadmap: Recap from xdc2025

By XDC 2026:

- Job prioritisation (CI-tron **done**  & LAVA **deemed good-enough**)
- Preemption support (CI-tron) **not done** 
- ci-collate expectations update from nightly runs

Roadmap: Recap from xdc2025

By XDC 2026:

- Job prioritisation (CI-tron **done**  & LAVA **deemed good-enough**)
- Preemption support (CI-tron) **not done** 
- ci-collate expectations update from nightly runs **WIP** 

Roadmap: Recap from xdc2025

By XDC 2026:

- Job prioritisation (CI-tron **done**  & LAVA **deemed good-enough**)
- Preemption support (CI-tron) **not done** 
- ci-collate expectations update from nightly runs **WIP** 
- Tools improvement (ci_run_n_monitor , ci-collate)

Roadmap: Recap from xdc2025

By XDC 2026:

- Job prioritisation (CI-tron **done**  & LAVA **deemed good-enough**)
- Preemption support (CI-tron) **not done** 
- ci-collate expectations update from nightly runs **WIP** 
- Tools improvement (ci_run_n_monitor **WIP** , ci-collate **WIP** )
 - ci_run_n_monitor-provided profiles to standardize processes (e.g. uprevs)

But this is not all we've done...

Quick summary of 2025-2026

- **Kept the system up and running** 🧑‍🚒 🔥
- Withstood LLM-crawlers DoS attacks
- Adapt CI to user requests

Quick summary of 2025-2026

- Kept the system up and running 🧑‍🚒 🔥
- Improved maintainability**
 - Filter out unwanted CI jobs at push time (doc)**
`git push -o ci.variables=hw_jobs=[kernel_drv, $mesa_drv, ...]`

Run new pipeline

Run for branch name or tag

main ▾

Inputs

Specify the input values to use in this pipeline. Any inputs left unselected will use their default values.

Name	Description
hw_jobs	List of hw job tags to restrict hardware testing to. Defaults to all .

Search

✓ all

none

-- mesa drivers --

anv

crocus

array

all



Quick summary of 2025-2026

- Kept the system up and running  
- **Improved maintainability**
 - Filter out unwanted CI jobs at push time
 - **Make the Mesa CI pipeline embeddable in other projects**
 - Needs “improvement” expectations mismatch to be acceptable before enforcing 
 - Pre-merge testing for our CI dependencies (linux, firmware, test suites, b2c, ...)
 - Mesa CI main embedded in: gfx-ci/linux , gfx-ci/firmware , piglit 

Quick summary of 2025-2026

- Kept the system up and running 🧑‍🚒 🔥
- **Improved maintainability**
 - Filter out unwanted CI jobs at push time
 - Make the Mesa CI pipeline embeddable in other projects
 - **Made almost all jobs use `gfx-ci/linux` & `gfx-ci/firmware`**

Quick summary of 2025-2026

- Kept the system up and running 🧑‍🚒 🔥
- Improved maintainability
- **Improved test coverage**
 - More devices tested: Imagination and Qualcomm
 - Android cuttlefish
 - Vulkan (ANV & RADV) and VA-API video
 - ASAN zink jobs on RADV

Quick summary of 2025-2026

- Kept the system up and running 🧑‍🚒 🔥
- Improved maintainability
- Improved test coverage
- **Reporting**
 - Trace rendering reports updated
 - Improved Marge pipeline summary for traces

Quick summary of 2025-2026

- Kept the system up and running 🧑‍🚒 🔥
- Improved maintainability
- Improved test coverage
- Reporting
- **Custom GitLab runner**
 - Better error handling and more relevant logs (CI-tron)
 - Goal: make all HW runners act like generic runners

Outline

Guiding principles

Practical challenges

What we've done

What now?

Annexes

What now?

👍 We have good guiding principles

What now?

👎 but we are not there yet

What now?



Let's fix the biggest pain points first!

Pain points - Expectation management, flakiness mitigations

Problems:

- Updating expectations is tedious, manual work
- Flakes are safe to add, but hard to safely remove

Pain points - Expectation management, flakiness mitigations

Problems:

- Updating expectations is tedious, manual work
- Flakes are safe to add, but hard to safely remove

Work in progress:

- Script to aggregate results from multiple jobs (ci-collate)

Pain points - Expectation management, flakiness mitigations

Problems:

- Updating expectations is tedious, manual work
- Flakes are safe to add, but hard to safely remove

Work in progress:

- Script to aggregate results from multiple jobs (ci-collate)

Future work:

- **Database of historical results for failures & flakes analysis**

Pain points - Expectation management, flakiness mitigations

Problems:

- Updating expectations is tedious, manual work
- Flakes are safe to add, but hard to safely remove

Work in progress:

- Script to aggregate results from multiple jobs (ci-collate)

Future work:

- Database of historical results for failures & flakes analysis
- **Run “stress tests” whenever possible, without affecting anyone (idle priority)**
 - Requires good job prioritization
 - Needs to be done for all Mesa jobs (including nightly)
 - Requires the ability to pause the long-running “idle” jobs (preemption)

Pain points - Long testing queues

Problem:

- Retrying slows down execution
 - Tests that time out take time away from useful tests
 - Retrying jobs that fail consistently slows down everyone

Pain points - Long testing queues

Problem:

- Retrying slows down execution
 - Tests that time out take time away from useful tests
 - Retrying jobs that fail consistently slows down everyone

Current solutions:

- Hosting more HW
- 1h timeout per MR

Pain points - Long testing queues

Problem:

- Retrying slows down execution
 - Tests that time out take time away from useful tests
 - Retrying jobs that fail consistently slows down everyone

Current solutions:

- Hosting more HW
- 1h timeout per MR

Future work:

- Job prioritization , preemption
- Consolidating similar jobs
- Shorter job & MR timeouts
- Pipelining MRs (start the next build while finishing testing)

Pain points - Reproducing CI jobs locally

Problem:

- Developers may need to reproduce the CI environment locally

Pain points - Reproducing CI jobs locally

Problem:

- Developers may need to reproduce the CI environment locally

Current solutions:

- Test environments are built as container images
- Cui-cui can run generic-runners jobs locally (not CI-tron, LAVA, nor Baremetal jobs)

Pain points - Reproducing CI jobs locally

Problem:

- Developers may need to reproduce the CI environment locally

Current solutions:

- Test environments are built as container images
- Cui-cui can run generic-runners jobs locally (not CI-tron, LAVA, nor Baremetal jobs)

Future work:

- Script to reproduce the test environment, stored in artifacts
- Make HW jobs look like generic-runners ones (no infra code in Mesa CI)

Pain points - Upreving dependencies

Problem:

- Keeping up with the development of our CI dependencies is time consuming

Pain points - Upreving dependencies

Problem:

- Keeping up with the development of our CI dependencies is time consuming

Current solutions:

- Manual uprev for Linux, linux-firmware, Khronos CTSes, VKD3D-Proton, ...
- `ci_run_n_monitor`-provided profiles to select groups of jobs (e.g. piglit jobs)
- `ci-uprev` for automatic uprev bot for Angle, Piglit, and VVL
- Automated testing of the Linux kernel in Mesa CI in `gfx-ci/linux`

Pain points - Upreving dependencies

Problem:

- Keeping up with the development of our CI dependencies is time consuming

Current solutions:

- Manual uprev for Linux, linux-firmware, Khronos CTSes, VKD3D-Proton, ...
- `ci_run_n_monitor`-provided profiles to select groups of jobs (e.g. piglit jobs)
- `ci-uprev` for automatic uprev bot for Angle, Piglit, and VVL
- Automated testing of the Linux kernel in Mesa CI in `gfx-ci/linux`

Future work:

- Expand `ci-uprev` to cover more projects
- Embed a Mesa CI pipeline in all the projects it depends on, enforcing non-regression.

Pain points - GitLab

Problem:

- Constant UI churn (e.g. renaming “issues” into “work items”), confusing users
- New and needed features broken in obvious ways, often never fixed
- Tons of regressions, then refusal to fix them
- Contempt for humans extended from users to employees (“replaced” with “AI”)
- Deploying GitLab locally to develop and test non-trivial changes is a PITA

Pain points - GitLab

Problem:

- Constant UI churn (e.g. renaming “issues” into “work items”), confusing users
- New and needed features broken in obvious ways, often never fixed
- Tons of regressions, then refusal to fix them
- Contempt for humans extended from users to employees (“replaced” with “AI”)
- Deploying GitLab locally to develop and test non-trivial changes is a PITA

Future work:

- We may want to look into Forgejo?

Pain points - GitLab

Known issues:

- CI:
 - Unable to select job(s) that we want to run, and automatically run the jobs that they depends on [[bug report](#)]
 - Unable to open a job log section when an error happened inside.
 - Unable to run CI jobs on each commit, only once per MR [[feature request](#)]
 - No job state/visual distinction between infrastructure failure (i.e. did not run the tests) and tests failure
 - Unable to fetch files from CI in private projects [[paid feature](#)]
 - Unable to reliably create only the relevant jobs for a branch, forcing us to always create huge pipelines [[bug report](#)]
 - Keyboard scrolling broken in several situations [[bug report](#)]
 - Unable to set job timeout dynamically (longer in forks than merge pipelines) [[feature request](#)]
 - Not allowed to use job artifacts for our needs (s3.fd.o workaround) [[paid feature](#)]
 - Job log web UI only shows the last 500 KiB of the job log, breaking rendering when e.g. several tests fail.
 - Unable to select job priority within a pipeline (e.g. run build-for-test jobs before build-only jobs).
 - Unable to select job priority across instance (e.g. merge jobs before non-merge jobs).
 - Unable to have separate artifacts for job success and job failure [[feature request](#)]
 - Unable to mark a job as non-blocking without ignoring its result [[feature request](#)]
 - Unable to display job artifacts in the browser (without downloading) [[bug report](#)] [[workaround](#)]
- Review:
 - Unable to see what changed during a rebase (range-diff) [[feature request](#)]
 - Unable to review commit messages [[feature request](#)]
 - Unable to get notified automatically on relevant MRs (CODEOWNERS) [[paid feature](#)]
 - Unable to ask reviews from multiple people [[paid feature](#)]
 - List of files changed (diffstat) missing from MR creation notification email [[feature request](#)]
 - Diff is incredibly slow to load ("rapid diff" is less slow, but still downloading one file at a time).
 - Unable to automatically link to the MR from the commit message [[feature request](#)]
- Misc:
 - MR/issues search is still lacking basics like and, or, not [[bug report](#)]
 - Unable to filter for MRs touching certain files/folders, or seeing which other in-flight MR touches them.
 - Unable to have groups like mesa/ci, mesa/intel, ... to tag people in MRs/issues.
 - Unable to disable tagging groups like @mesa which tags several hundreds of people.

Roadmap

By XDC 2027:

- **ci-collate: automate expectations updates**
 - Integrate in nightly runs.

Roadmap

By XDC 2027:

- ci-collate: automate expectations updates
- **ci_run_n_monitor: efficiently run our CI jobs**
 - Integrate ci-collate to simplify expectations updates.
 - Support delegating job management to GitLab.
 - Allow multiple users to run different jobs in the same pipeline.

Roadmap

By XDC 2027:

- ci-collate: automate expectations updates
- ci_run_n_monitor: efficiently run our CI jobs
- **Make it trivial to reproduce a CI job on a local machine**

Roadmap

By XDC 2027:

- ci-collate: automate expectations updates
- ci_run_n_monitor: efficiently run our CI jobs
- Make it trivial to reproduce a CI job on a local machine
- **Embed Mesa CI pipelines in other projects**
 - gfx-ci/firmware
 - drm/amd, drm/msm, ...
 - mesa/piglit

Roadmap

By XDC 2027:

- ci-collate: automate expectations updates
- ci_run_n_monitor: efficiently run our CI jobs
- Make it trivial to reproduce a CI job on a local machine
- Embed Mesa CI pipelines in other projects
- **Improve CI monitoring (statistics page, notifications) in a farm-agnostic way**

Roadmap

By XDC 2027:

- ci-collate: automate expectations updates
- ci_run_n_monitor: efficiently run our CI jobs
- Make it trivial to reproduce a CI job on a local machine
- Embed Mesa CI pipelines in other projects
- Improve CI monitoring (statistics page, notifications) in a farm-agnostic way
- **DRM CI 2.0? Talk to us if this is something you care about!**

Roadmap

By XDC 2027:

- ci-collate: automate expectations updates
- ci_run_n_monitor: efficiently run our CI jobs
- Make it trivial to reproduce a CI job on a local machine
- Embed Mesa CI pipelines in other projects
- Improve CI monitoring (statistics page, notifications) in a farm-agnostic way
- DRM CI 2.0? Talk to us if this is something you care about!
- **Preemption support (CI-tron) → 100% DUT usage doesn't delay merge jobs**

Did we miss anything? Let us know!

- Label ~"CI pain point" to review and report more issues.

Did we miss anything? Let us know!

- Label ~"CI pain point" to review and report more issues.
- Workshop tomorrow morning at 9:10-10:40 am:

Mesa CI - What still needs to be done, and how to get there?

Did we miss anything? Let us know!

- Label ~"CI pain point" to review and report more issues.
- Workshop tomorrow morning at 9:10-10:40 am:

Mesa CI - What still needs to be done, and how to get there?

- Join the team, we hand out a lot of (Anubis) cookies!

Outline

Guiding principles

Practical challenges

What we've done

What now?

Annexes

ci_run_n_monitor.sh

ci_run_n_monitor.sh

- Can point to a specific pipeline, or a merge request (latest pipeline), or a commit (searched in the user's fork or mesa/mesa).
- Will run the jobs requested, and skip the rest (saving those resources for other users).
 - Prints what will be triggered, what's running, and the results summary when complete.
- Can be used to stress-test (running jobs multiple times).
 - With a time limit, and eventually include a results collate.
- It's experimental on other projects.
- Improvements in progress.

CI infrastructures

Baremetal

- The original solution for Mesa CI (thanks Emma <3)
- All the test machines boot sequencing stored in Mesa CI
- Minimal software requirements on the host:
 - GitLab runner
 - NFS
 - NGINX caching proxy
- Not developed anymore, slated for removal
- Cons:
 - Depend on code stored in Mesa
 - No interactive access to test machines
 - No sharing of machines across forges

CI infrastructures

LAVA

- Infrastructure created by Linaro, mostly used for Linux kernel testing
- Support added by Collabora to test Mesa on the same machines as Kernel CI
- Allows to share the DUTs with other CIs
- Options for booting, rootfs overlays, log propagation
- Actively developed
- Cons:
 - Depend on code stored in Mesa
 - No interactive access to test machines
 - Using `lava-job-submitter` script instead of an adapted `gitlab-runner`.
 - Overbooking possible, so jobs starting without DUT available, consuming running time, leading to timeouts and jobs failing.

CI infrastructures

CI-tron

- Infrastructure funded by Valve to address the structural issues of the other solutions
- Aiming to be as maintainable and easy to use as possible so that developers can expose their test machines on fd.o.
- Benefit of hindsight on many existing HW CI systems: Intel Mesa/GFX CI, LAVA, Baremetal, EzBench, ...
- Actively developed
- Cons:
 - Newer, not as mature yet (e.g. job format not finalized)
 - Not everything we want is implemented yet, but already has everything that LAVA has
 - Limited documentation

ci-collate

ci-collate

- Tool to automate the expectations update from a give pipeline.
 - Features to work with a single job or one pipeline.
 - But Mainly to prepare a patch with the expectation changes.
 - Usage:
 - `ci-collate job [--trace|--artifact PATH] -- JOB_ID`
 - `ci-collate pipeline --job-filter REGEX [REGEX ...] -- PIPELINE_ID`
 - `ci-collate patch --jobs REGEX [REGEX ...] -- PIPELINE_ID`
- By default, working only with `failures.csv`.
 - Can use `results.csv` on demand with important impact on memory usage.
 - Eventually `depq-runner` could provide a `flakes.csv`.
- Integration within `ci_run_n_monitor` 🚧.

ci-uprev

ci-uprev

- Tool to regularly **UP**date **REV**ision in use for a dependency in our projects.
- Initially a script that became larger and larger.
- Hard to scale at the current point:
 - uprev ANGLE in Mesa,
 - uprev Mesa in Virglrenderer,
 - uprev Piglit in Mesa,
 - uprev VVL in Mesa.
- Refactor in progress 🚧
 - Decouple code from specific uprevs.
 - Use a **yaml** to define the behaviour.
 - Multiple dependencies in one uprev.
 - Use ci_run_n_monitor and ci-collate.