# Continuous Integration Service at Fermilab

**Vito Di Benedetto for CI Project**
Vladimir Podstavkov, Michele Fattoruso, Bruno Coimbra
Fermilab

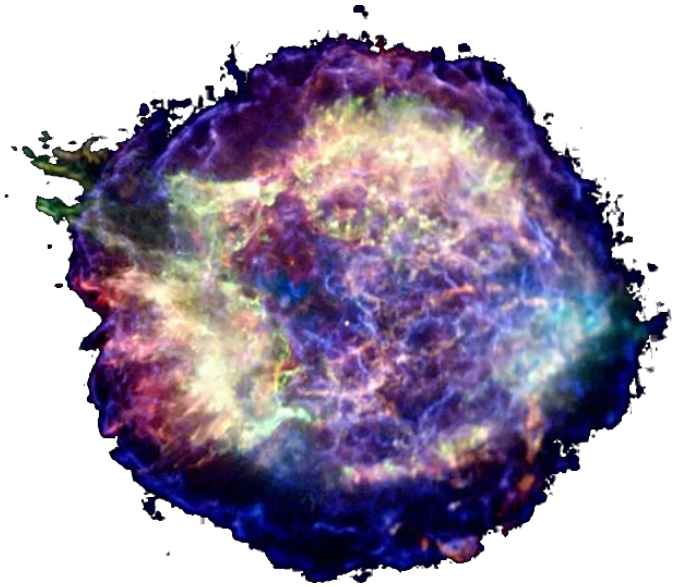
CHEP 2018 July 12, 2018

# Why we care about the Continuous Integration

- All developers write code following good practice… and they test their code before committing it

# Why we care about the Continuous Integration

- All developers write code following good practice… and they test their code before committing it
  - but sometime the code:
    - blows up at build time
    - crashes at run time
    - when used in production trashes down computing resources



# Why we care about the Continuous Integration

- All developers write code following good practice… and they test their code before committing it
  - but sometime the code:
    - blows up at build time

Yeah, in the meantime
module “A” has been updated to vX.Y
and my code doesn’t fit anymore

# Why we care about the Continuous Integration

- All developers write code following good practice… and they test their code before committing it
  - I need more info for my analysis
  - I need to use this package to get a more accurate simulation

# Why we care about the Continuous Integration

- All developers write code following good practice… and they test their code before committing it

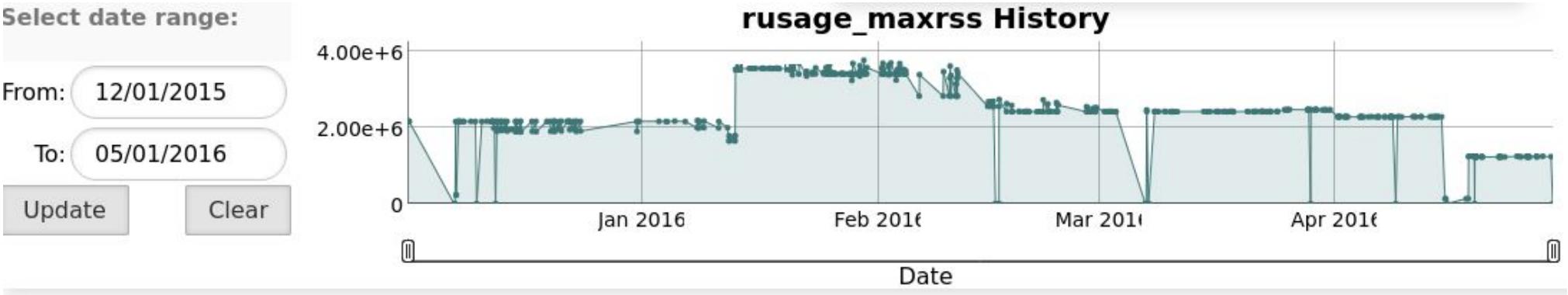
I need more info for my analysis

I need to use this package to get a more accurate simulation

the memory footprint doesn't fit my system anymore

# Why we care about the Continuous Integration

- All developers write code following good practice… and they test their code before committing it




the memory footprint doesn't fit my system anymore



Fermilab

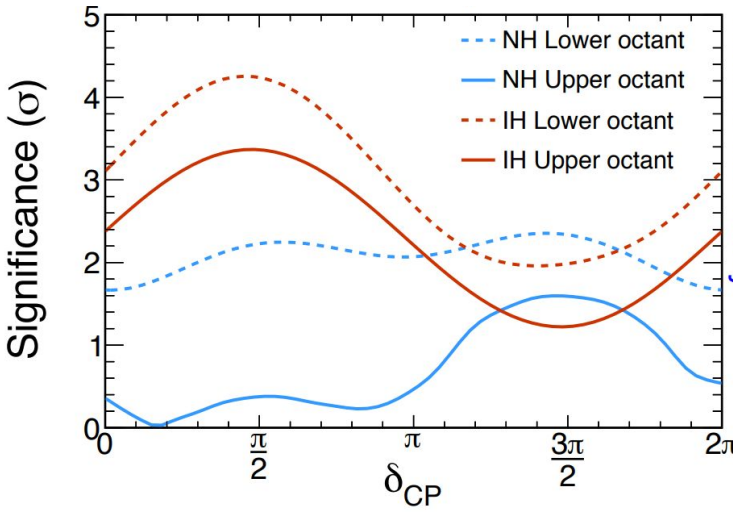
# Why we care about the Continuous Integration

- All developers write code following good practice… and they test their code before committing it
  - A good testing strategy that provides rapid feedback helps to identify defects introduced by code changes quickly
  - Issues detected early on in development are typically smaller, less complex and easier to resolve

# Why we care about the Continuous Integration

- All developers write code following good practice… and they test their code before committing it
  - A good testing strategy that provides rapid feedback helps to identify defects introduced by code changes quickly
  - Issues detected early on

…and physicist can spend more time on analysis

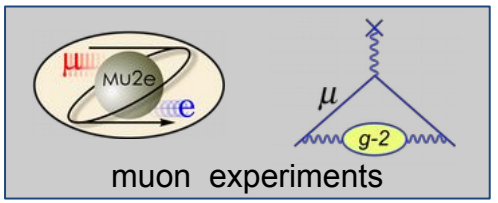


# FIFE CI Goals

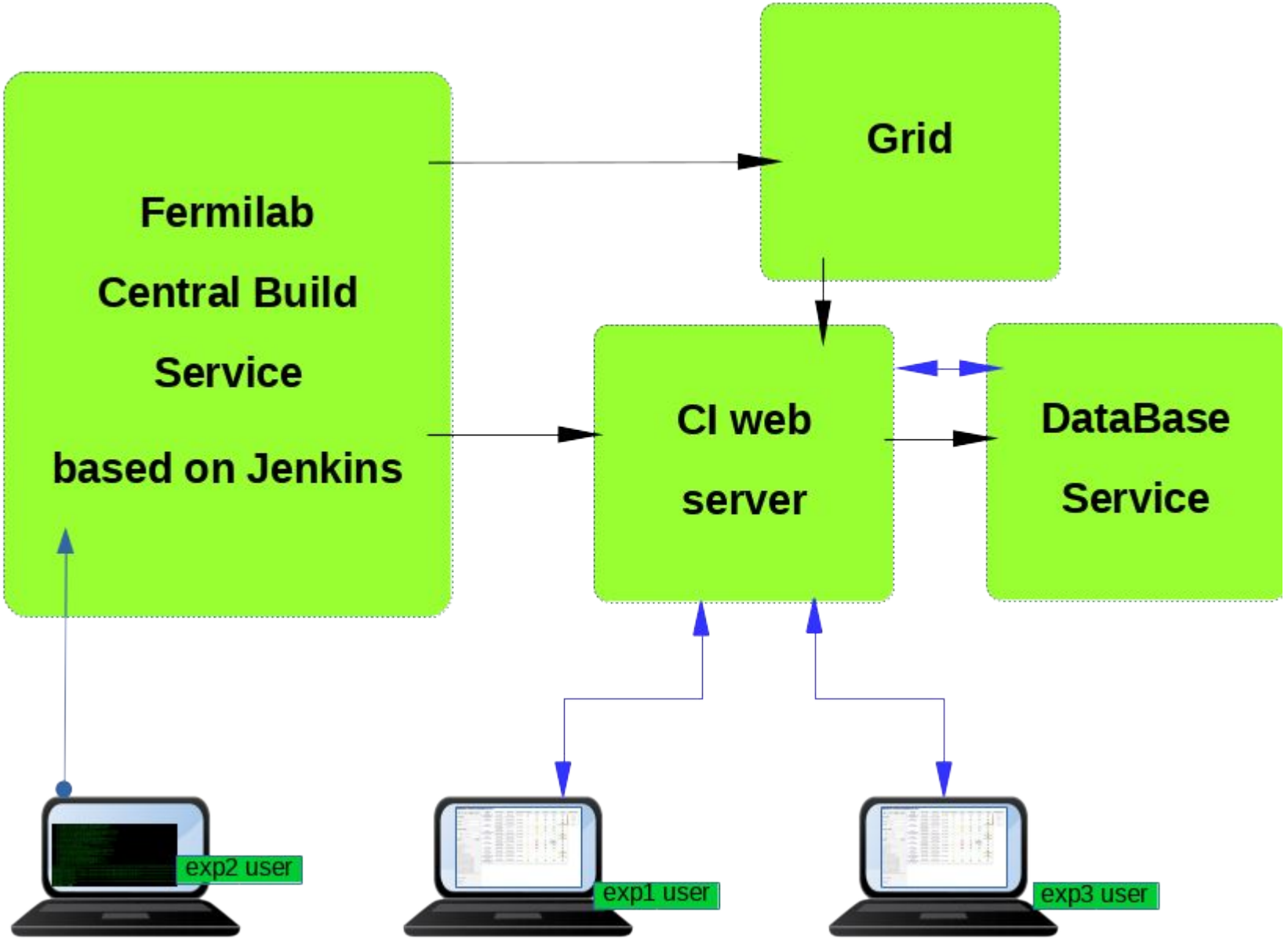
- Provide a common framework for all FIFE experiments and projects to build and test their software.
- Offer an access to comprehensive testing from unit tests to physics validation through regression test.
- Allow to trace code revision and resource usage stats for every build.
- CI builds can be triggered:
  - on demand by users
  - by commit in monitored repositories
  - periodically (cron)
- Provide UI that clearly indicates issues with current release
- Enable automatic notification in case of failures.

# CI: It is Complicated ...

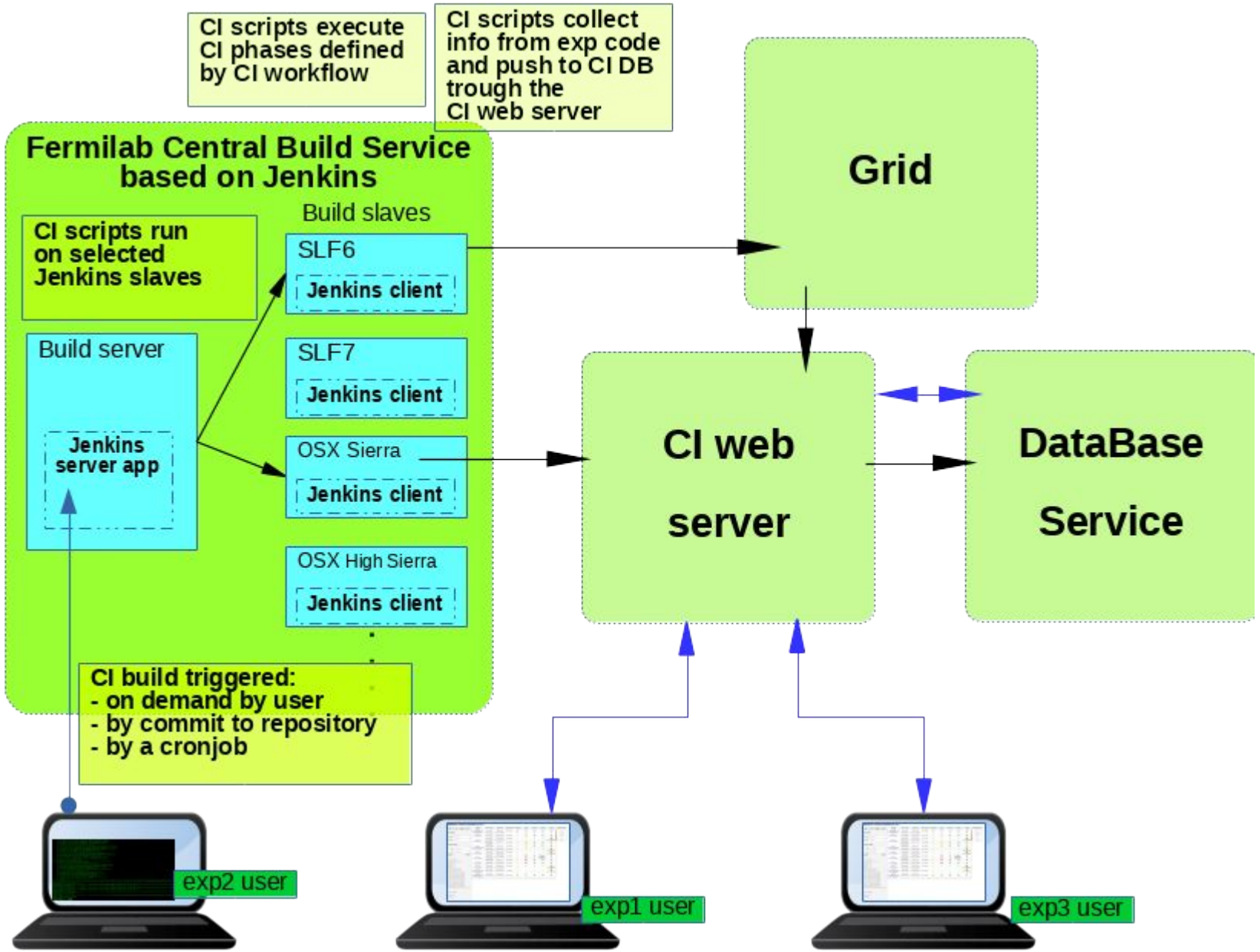
- The CI is used by a wide variety of experiments and projects
- Build systems:
  - CMake
  - Make
  - Custom one (MRB, SRT, scons)
- Version Control Systems:
  - CVS
  - SVN
  - GIT
  - Mercurial
- Repository is hosted:
  - Fermilab Redmine and/or elsewhere
- Code is tested on:
  - SL6/7
  - Mac OSX

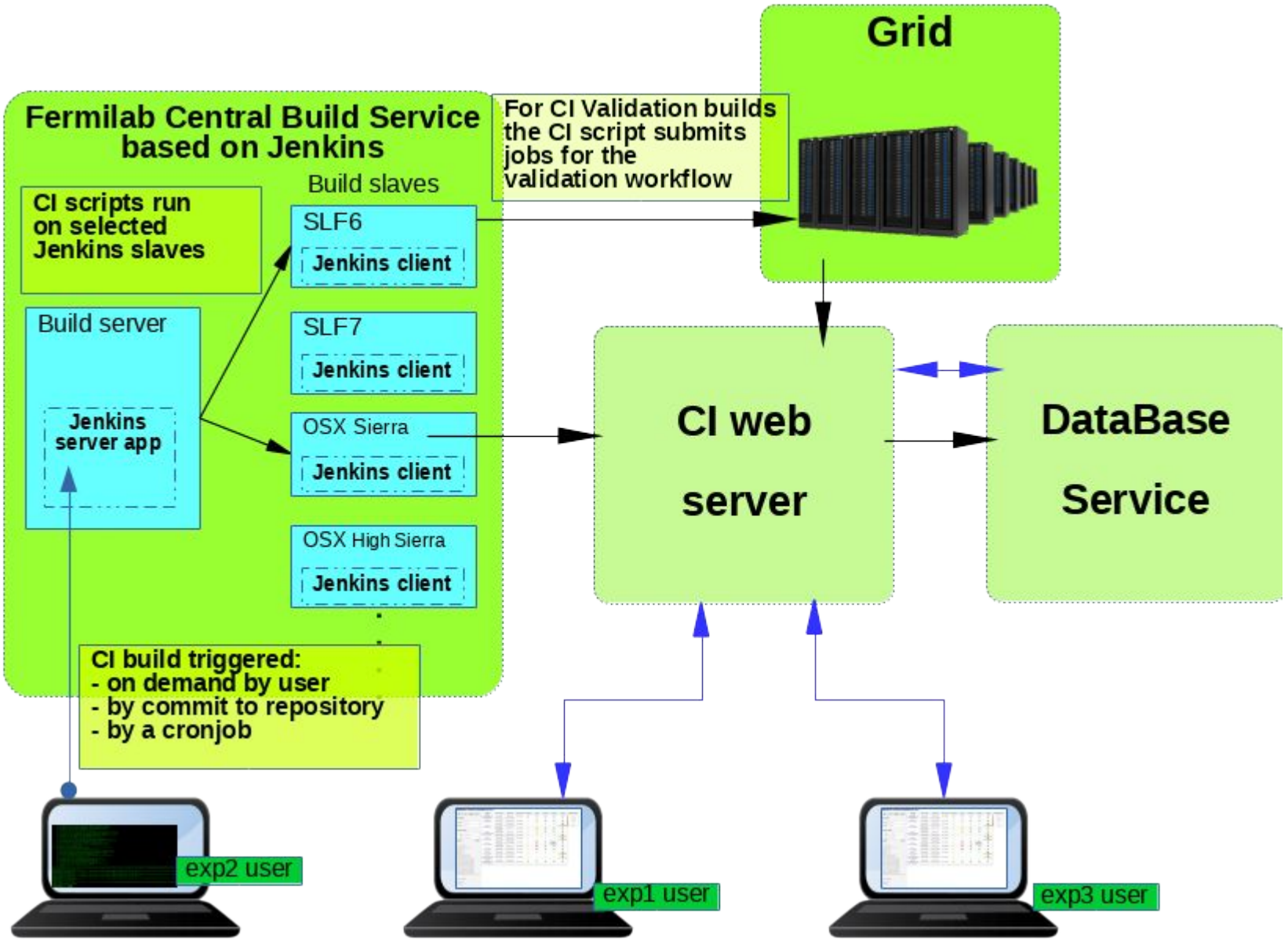

# CI Architecture



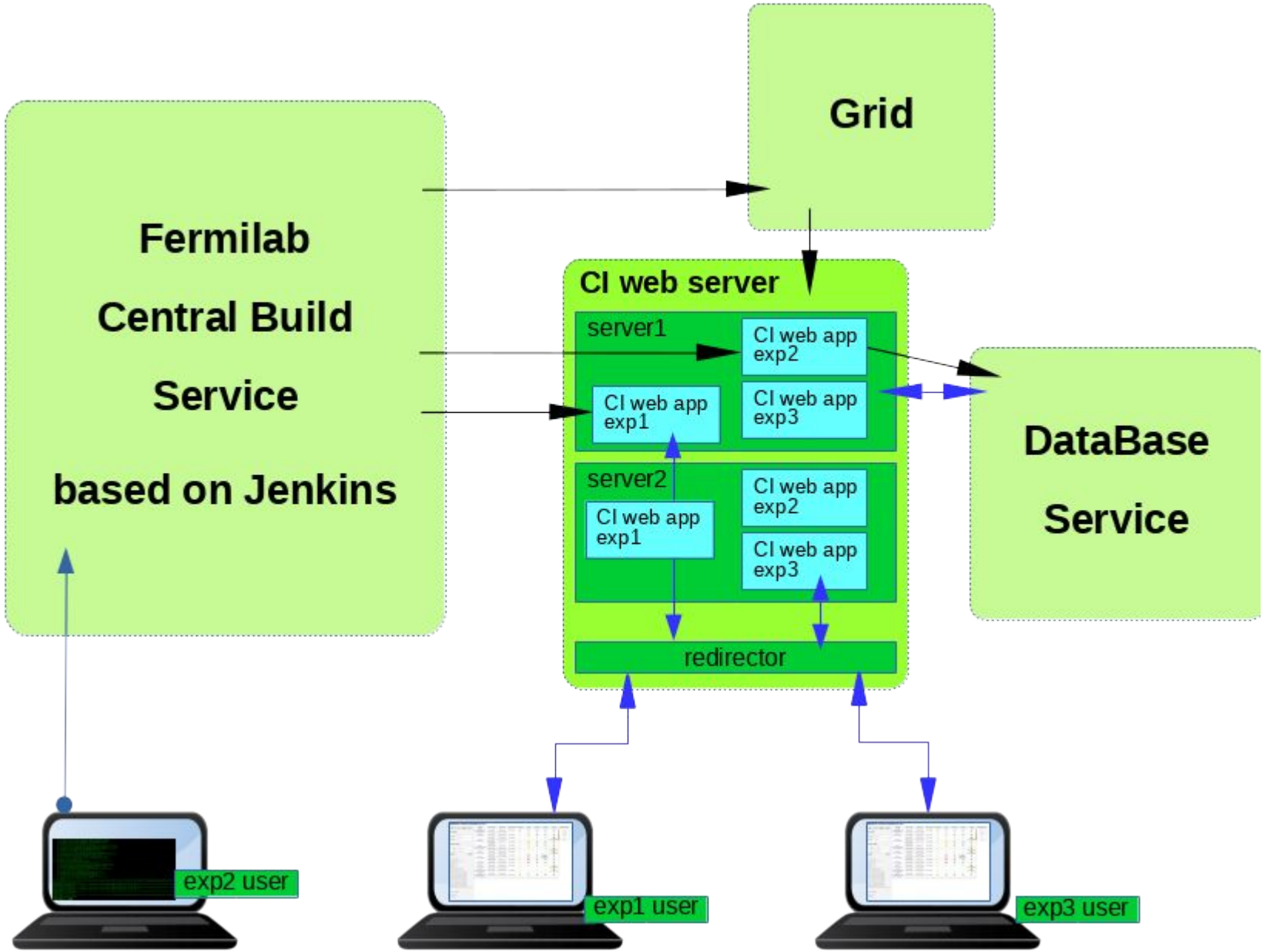
# CI Architecture: Build service



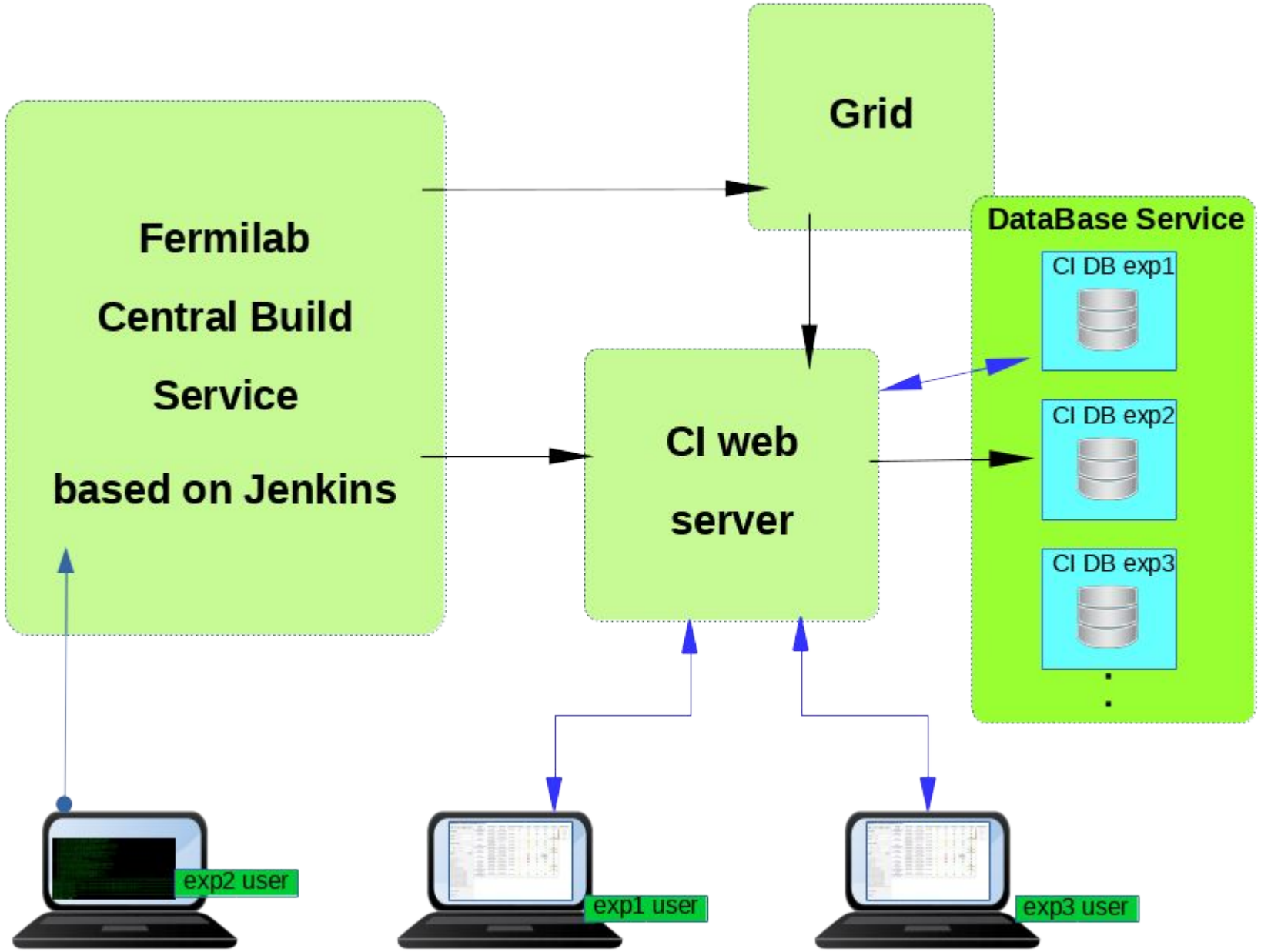
# CI Architecture: Build service



# CI Architecture: CI web server



# CI Architecture: DataBase

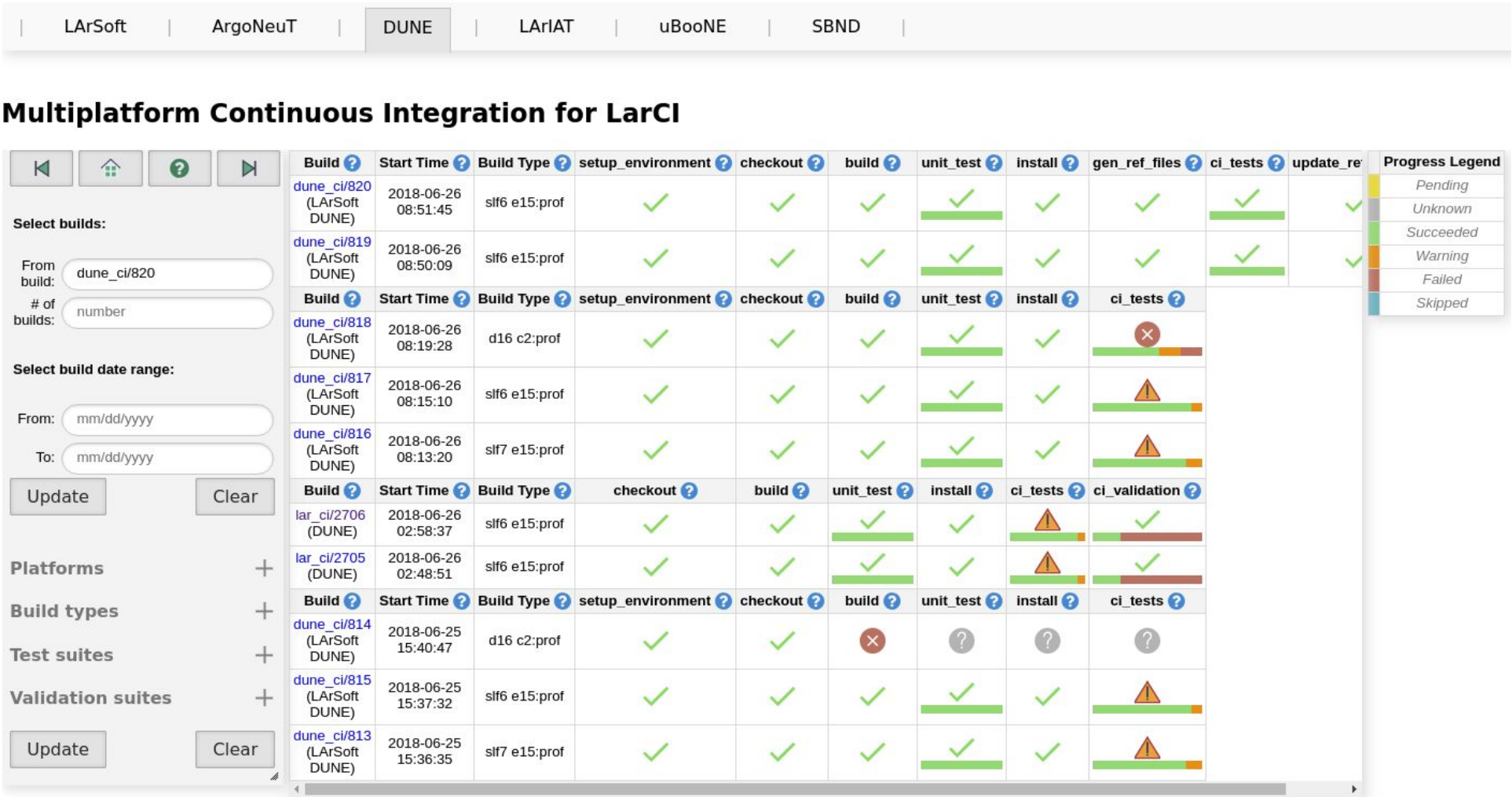


# CI workflow configuration

- The workflow configuration allow the user to run an “almost” arbitrary procedure.
- Workflow consists of multiple steps (phases) executed sequentially:
  - env setup
  - checkout
  - build
  - unit tests
  - install
  - regression tests
  - validation tests
- During some phases tasks could be executed in parallel
- For each phase users can define:
  - initialization and finalizing procedures
  - work to do
  - logs to report
  - the format of the report generated from the logs

# CI dashboard overview

## CI dashboard used by LArSoft and LArSoft-based experiment code

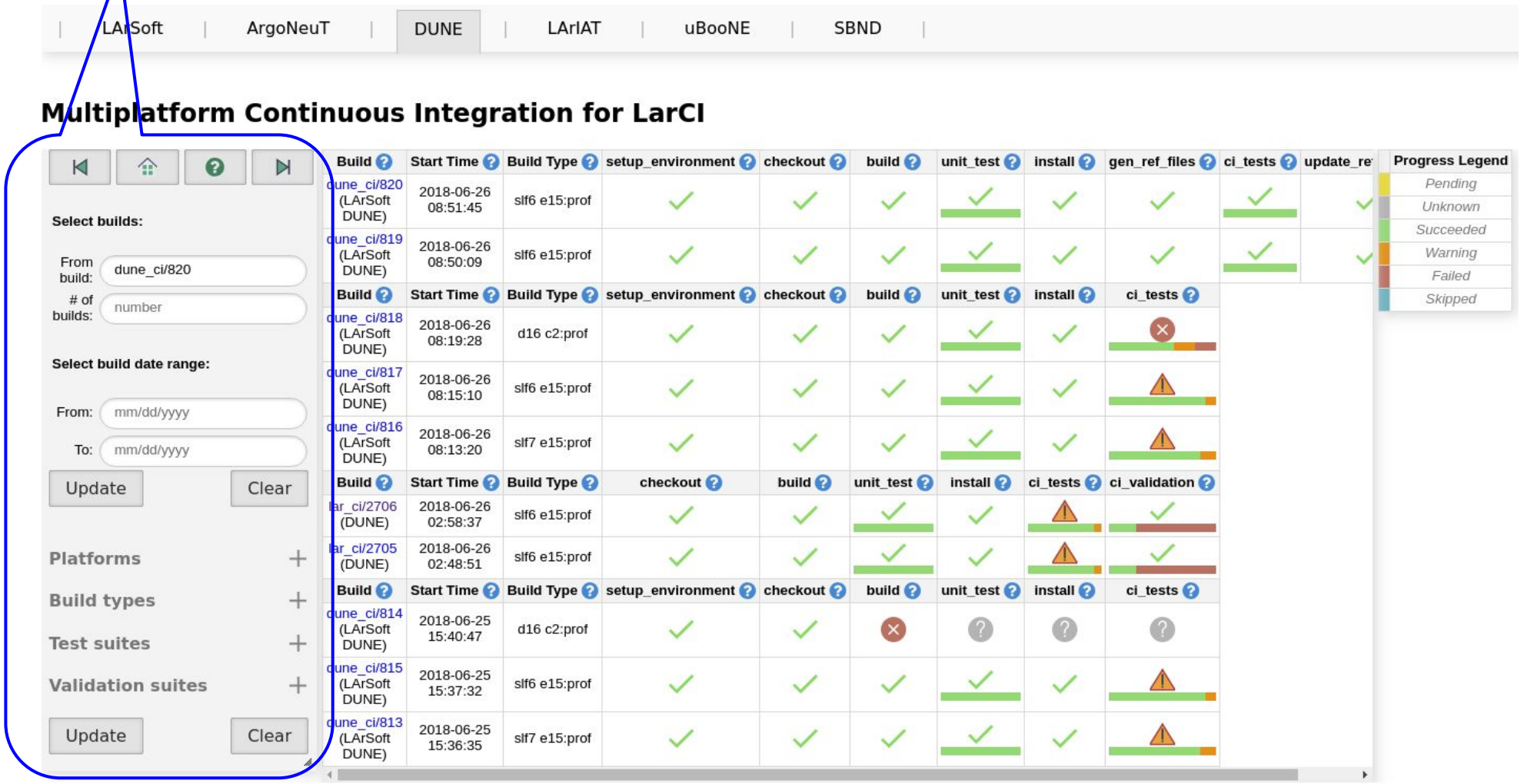





Fermilab

# CI dashboard details

This panel allows to filter CI builds using different criteria

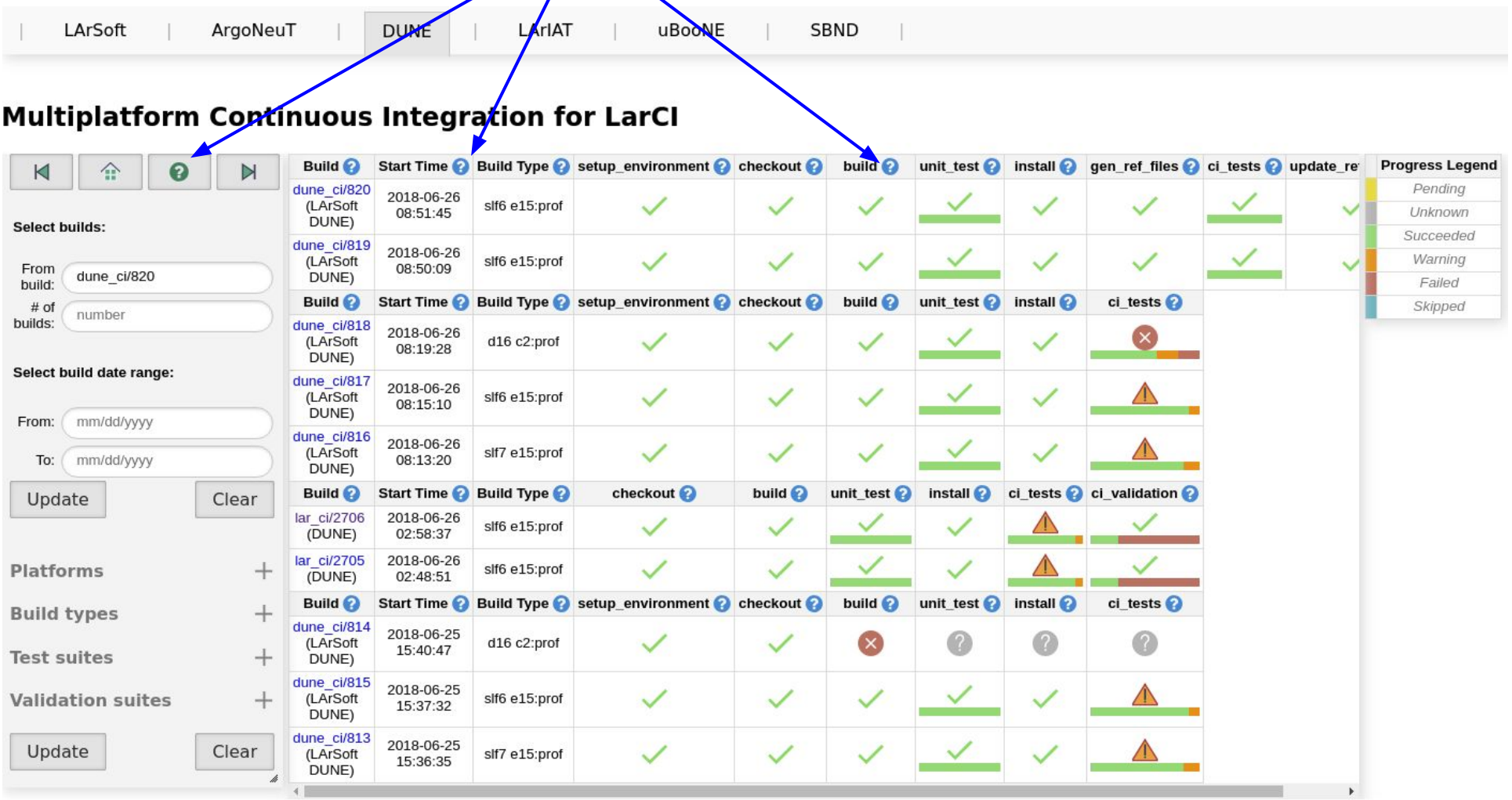





Fermilab

# CI dashboard details

(?) markers give access do wiki documentation

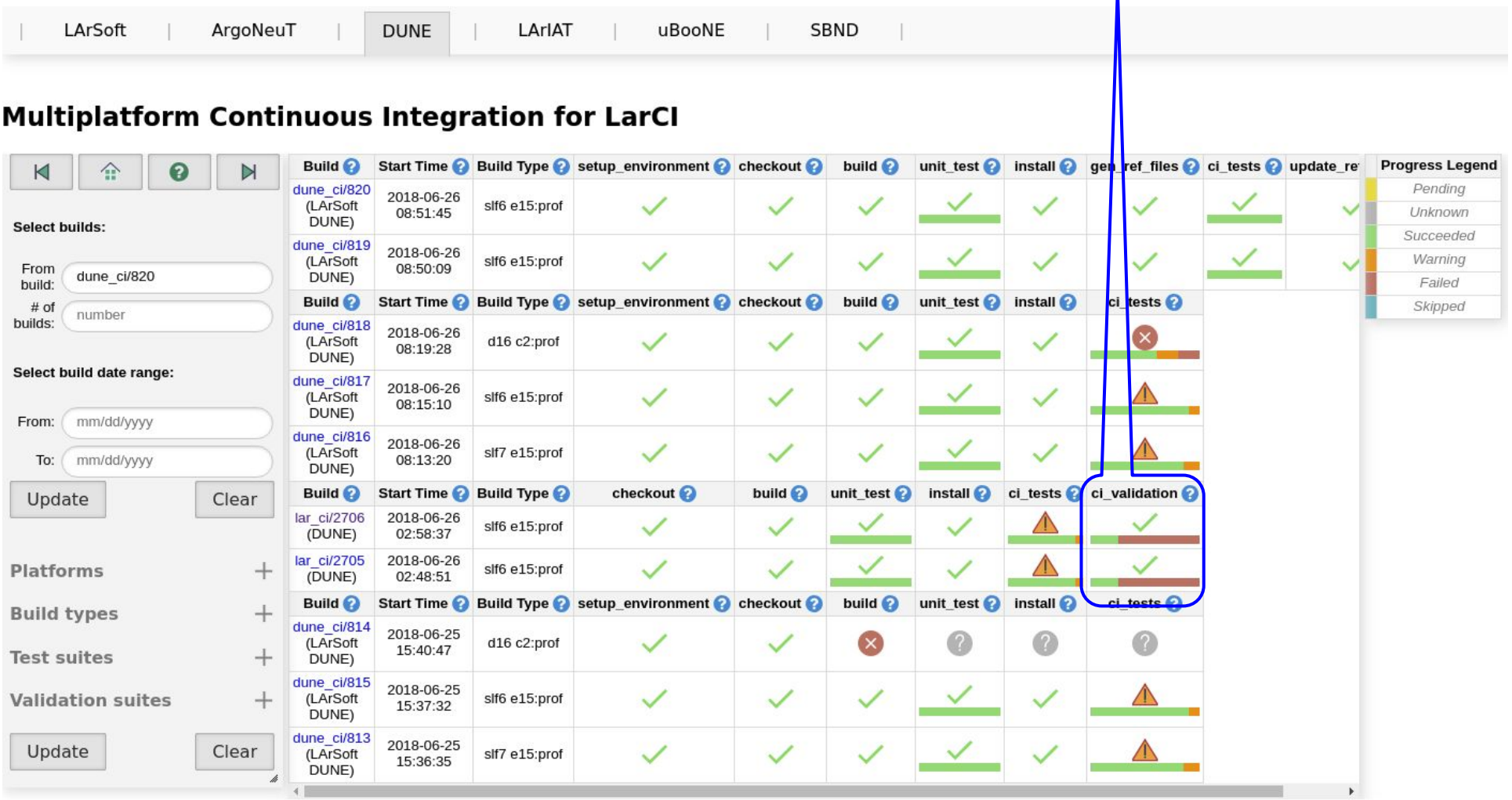





Fermilab

# CI dashboard details

CI Validation reports some failed jobs but validation results are OK

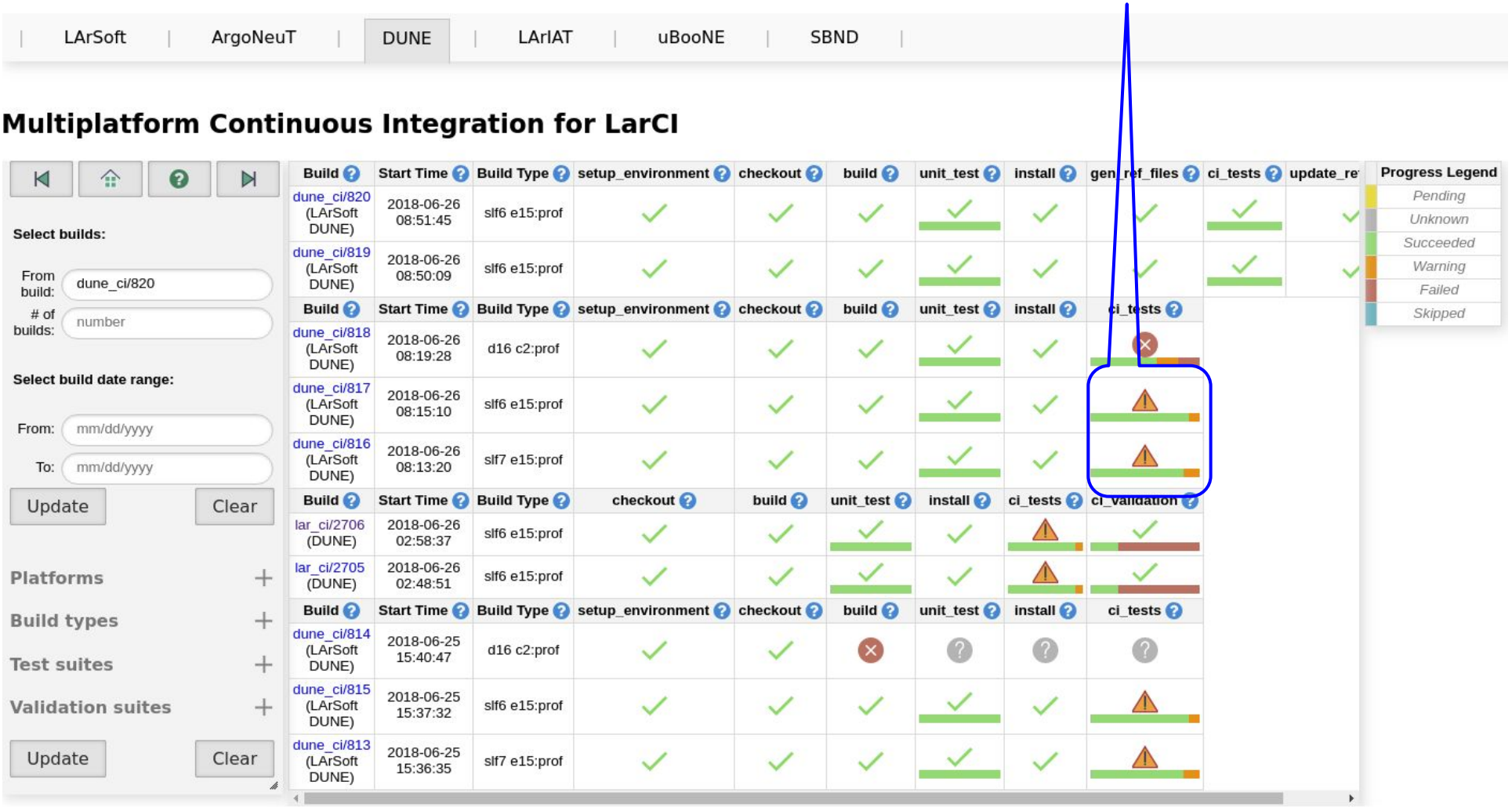





Fermilab

# CI dashboard details

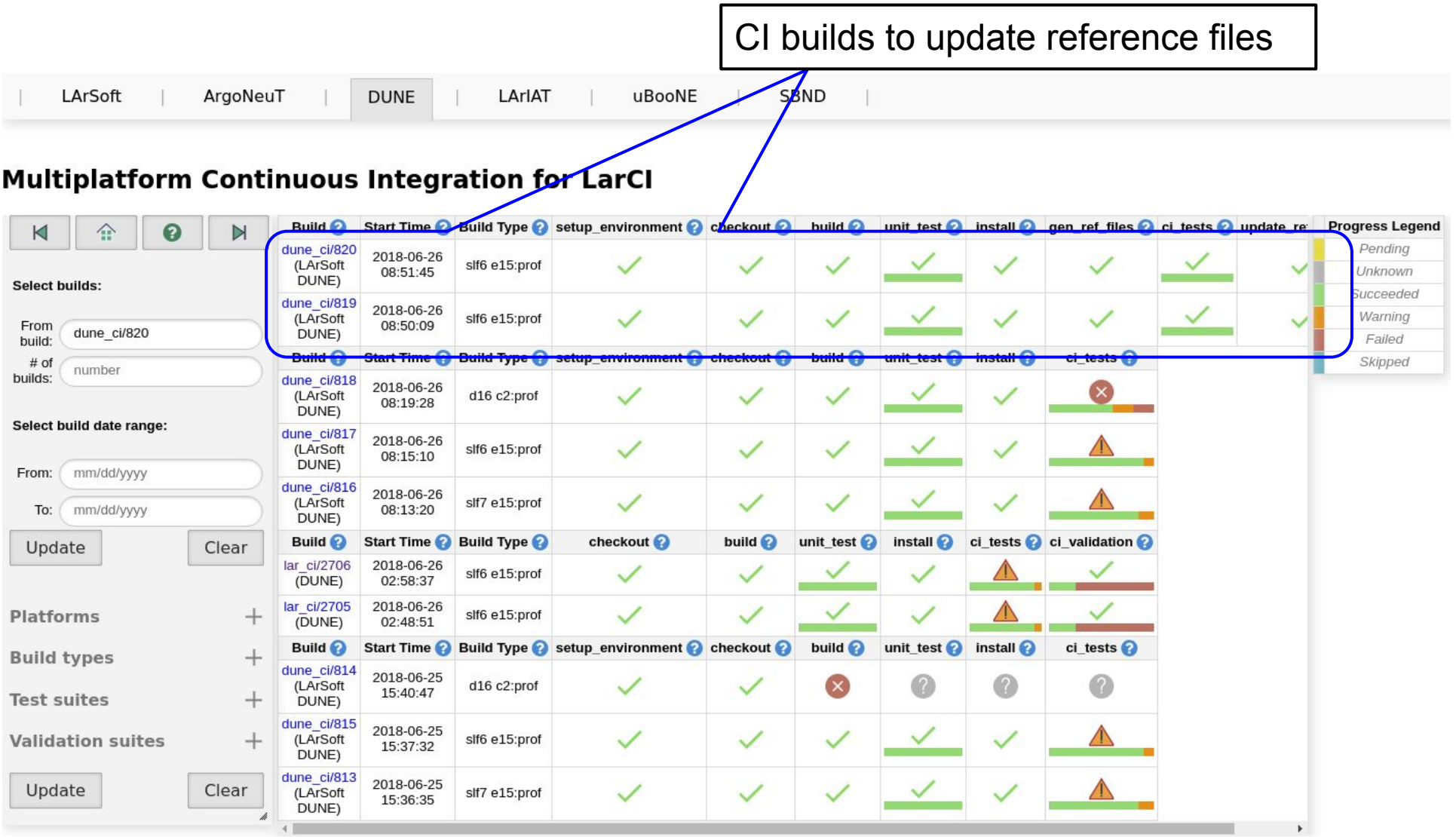
Warning in regression tests notifies that some stage output is changed






Fermilab

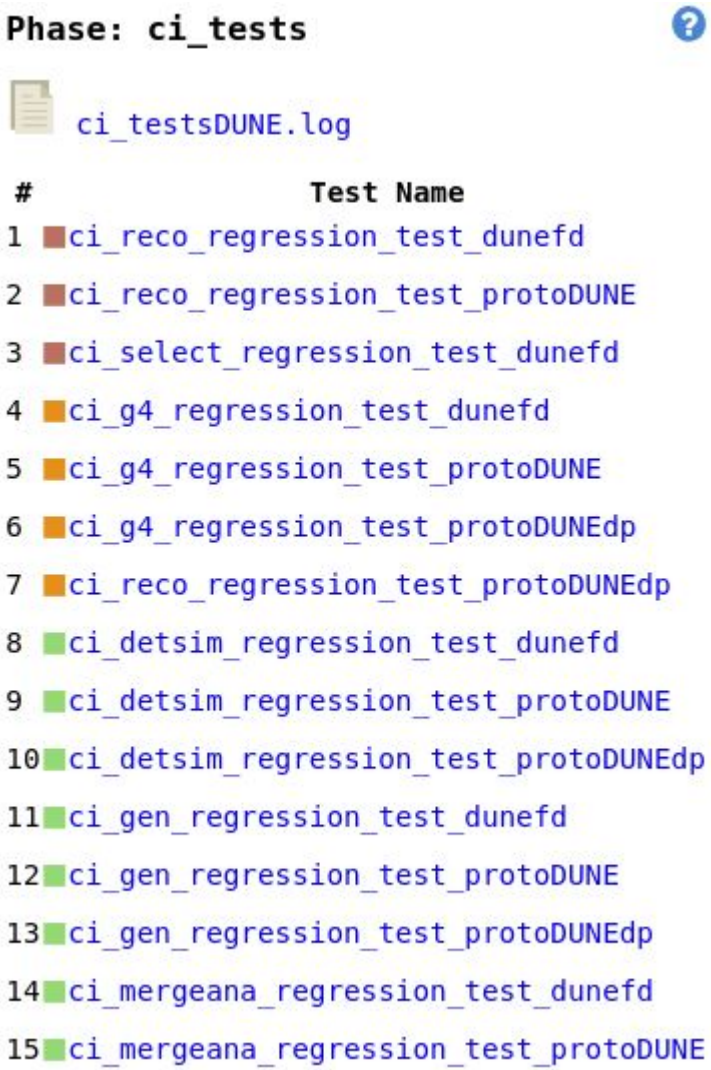
# CI dashboard details





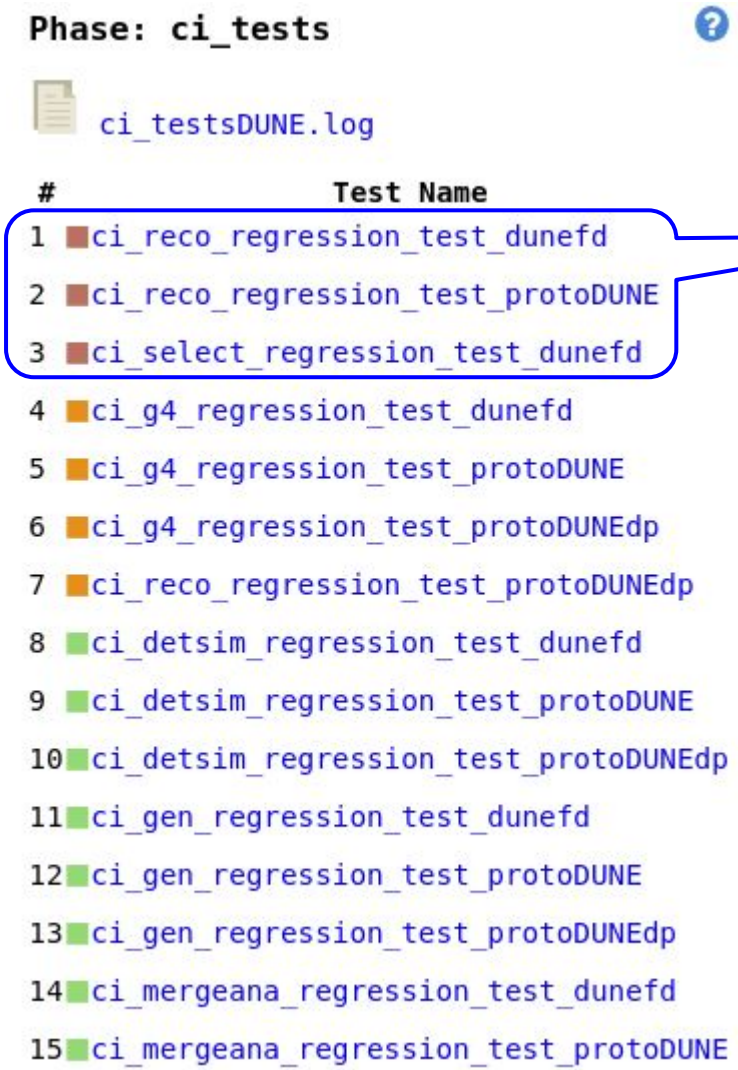

Fermilab

# CI (regression) tests status



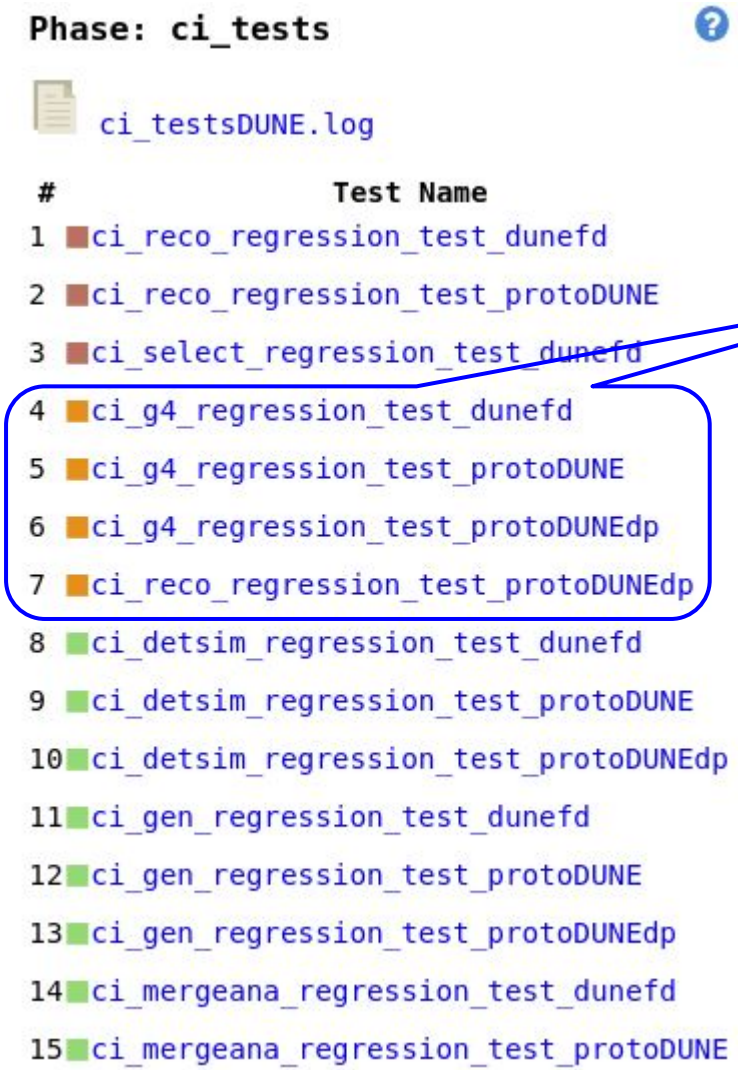

- CI tests are sorted according to the status severity:
  - failure (red)
  - warning (orange)
  - successful (green)

# CI (regression) tests status




- CI tests are sorted according to the status severity:
  - failure (red)
    - problem to run the code
    - in case of crash there is the backtrace extracted from the coredump

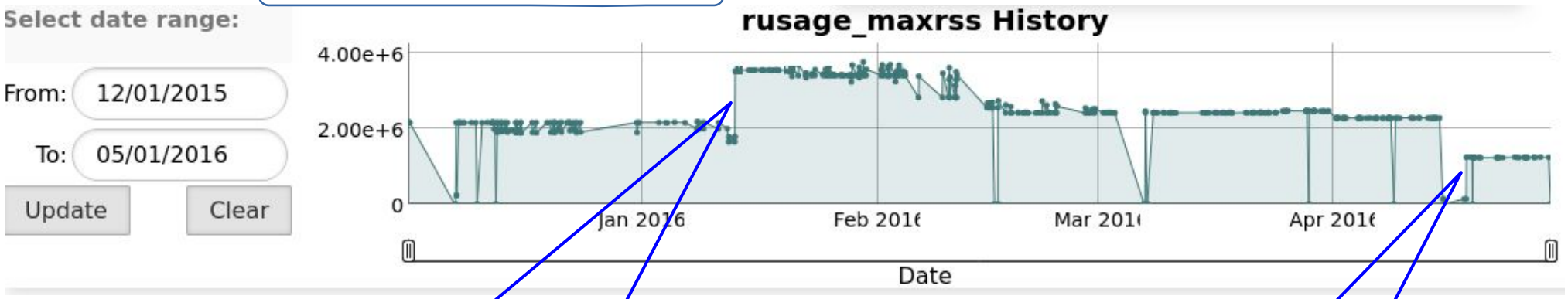
# CI (regression) tests status




- CI tests are sorted according to the status severity:
  - failure (red)
  - warning (orange)
    - to code runs, but the output is different from the reference

# CI (regression) tests: memory usage

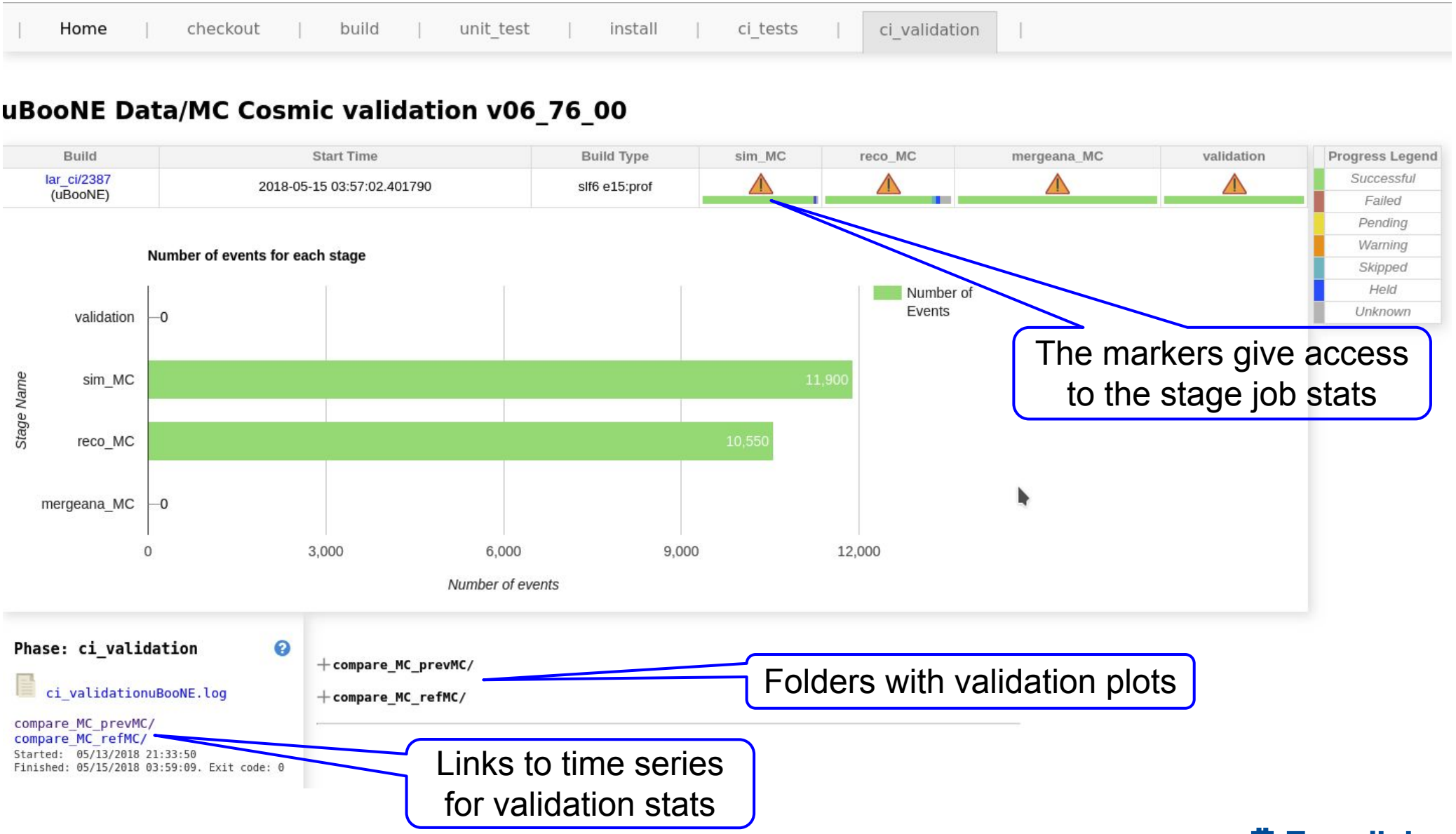
Memory usage for 1 event




By switching from CRY to CORSICA as cosmic shower generator, the memory usage jumped from ~2Gb to ~3.5Gb

After memory optimization the memory usage went down to 1.2Gb.

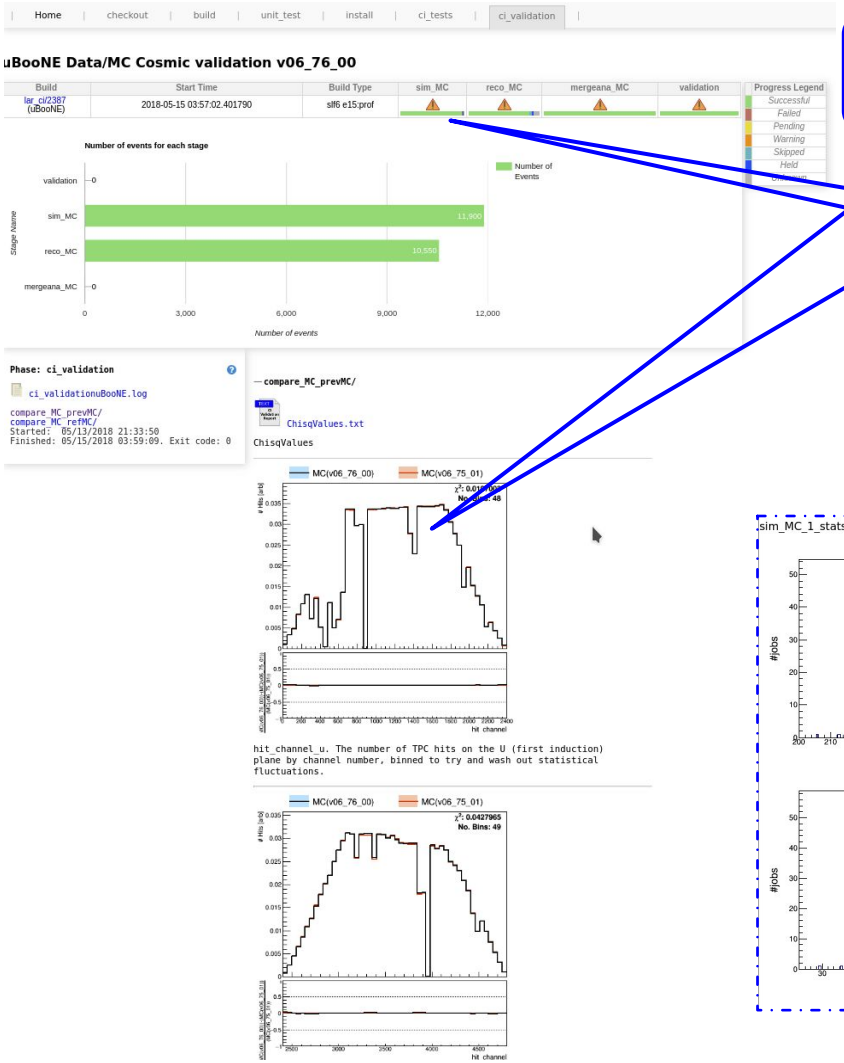
# CI Validation dashboard





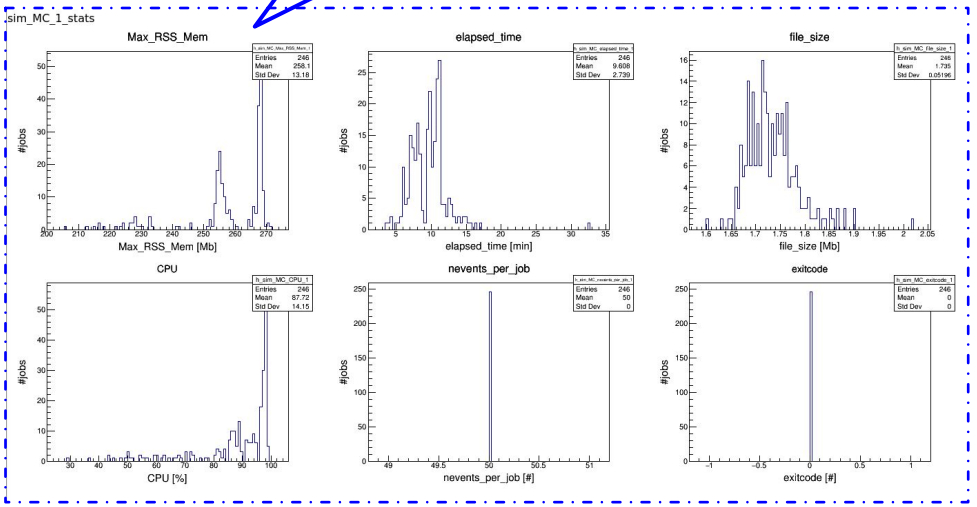

Fermilab

# CI Validation dashboard




Folders with validation plots expanded

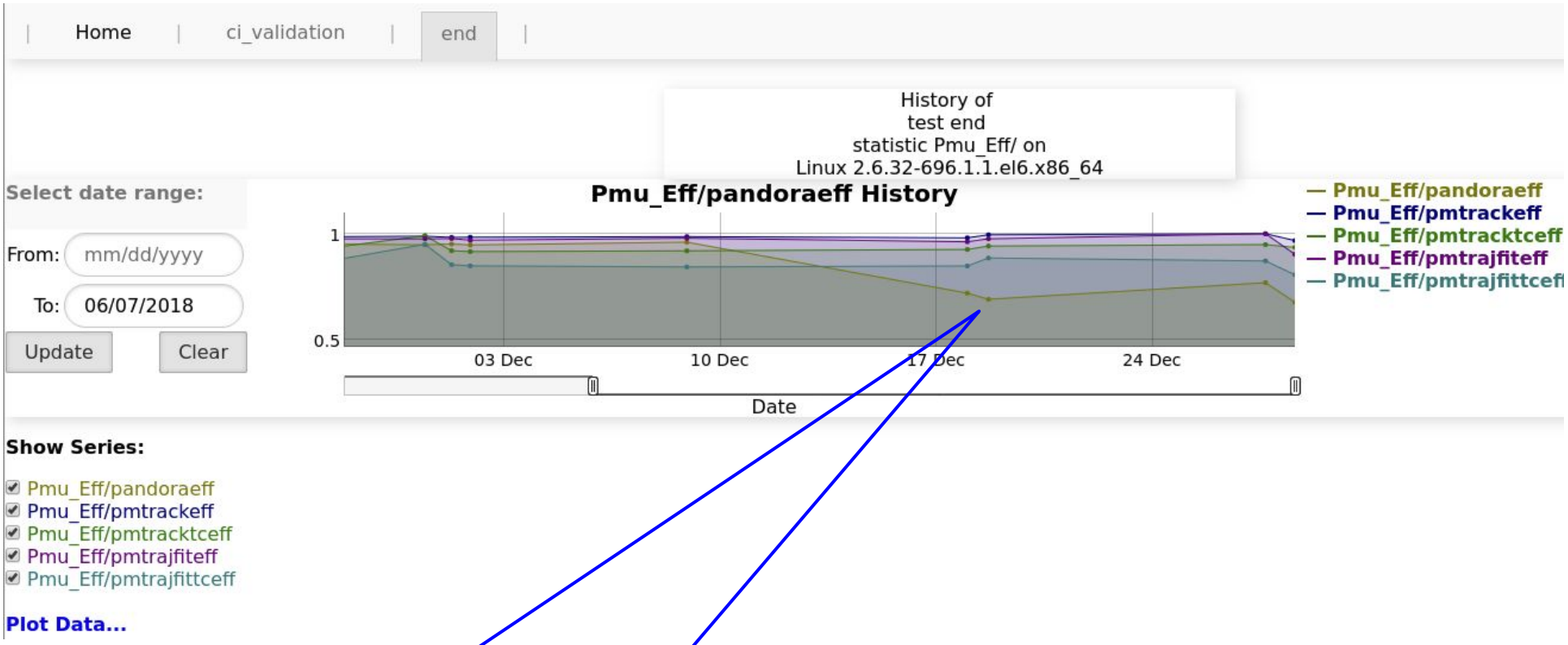
The *stage* marker gives access to job resource stats






Fermilab

# CI Validation: efficiency time series




The CI Validation reported a significant drop in the track reconstruction efficiency for one of the reconstruction algorithms

# Summary

- The FIFE CI service provides a comprehensive framework for FIFE experiments and projects to test and validate their offline production and analysis code on the supported platforms.
  - It allows to validate experiment code through grid jobs.
  - Provides a dashboard for easy access to logs and statistical graphs.
- Currently the CI service is in use by the ArgoNeuT, DUNE, g-2, LArIAT, MINERvA, mu2e, NOvA, SBND and uBooNE experiments and by the following projects: ART and LArSoft software suites, GENIE, GlideinWMS and Rivet.
- The service proved to be very useful to intercept issues in experiments' code early on and get it fixed before running production jobs.
- Future development plans include:
  - enhanced physics validation features
  - support for code profiling

# Thank you for your attention