Sandia National Laboratories

Exceptional service in the national interest

TRILINOS CI TESTING/CONTRIBUTION OVERVIEW

Samuel E. Browne

Sandia National Laboratories

HPSF Conference 05/07/2025

U.S. DEPARTMENT OF ENERGY | NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

TRILINOS DEVSECOPS TEAM

The Trilinos DevSecOps (Framework) team is responsible for the CI processes that enable the assurance of software quality for the Trilinos code suite. It also functions to assist with work related to the build system of this large and complex C++ suite.

With the transition of Trilinos to the HPSF, open-source partner contribution and collaboration is more greatly emphasized. This increased emphasis highlights the need to improve the contribution experience, particularly with respect to accessible CI processes, and simpler build experience.

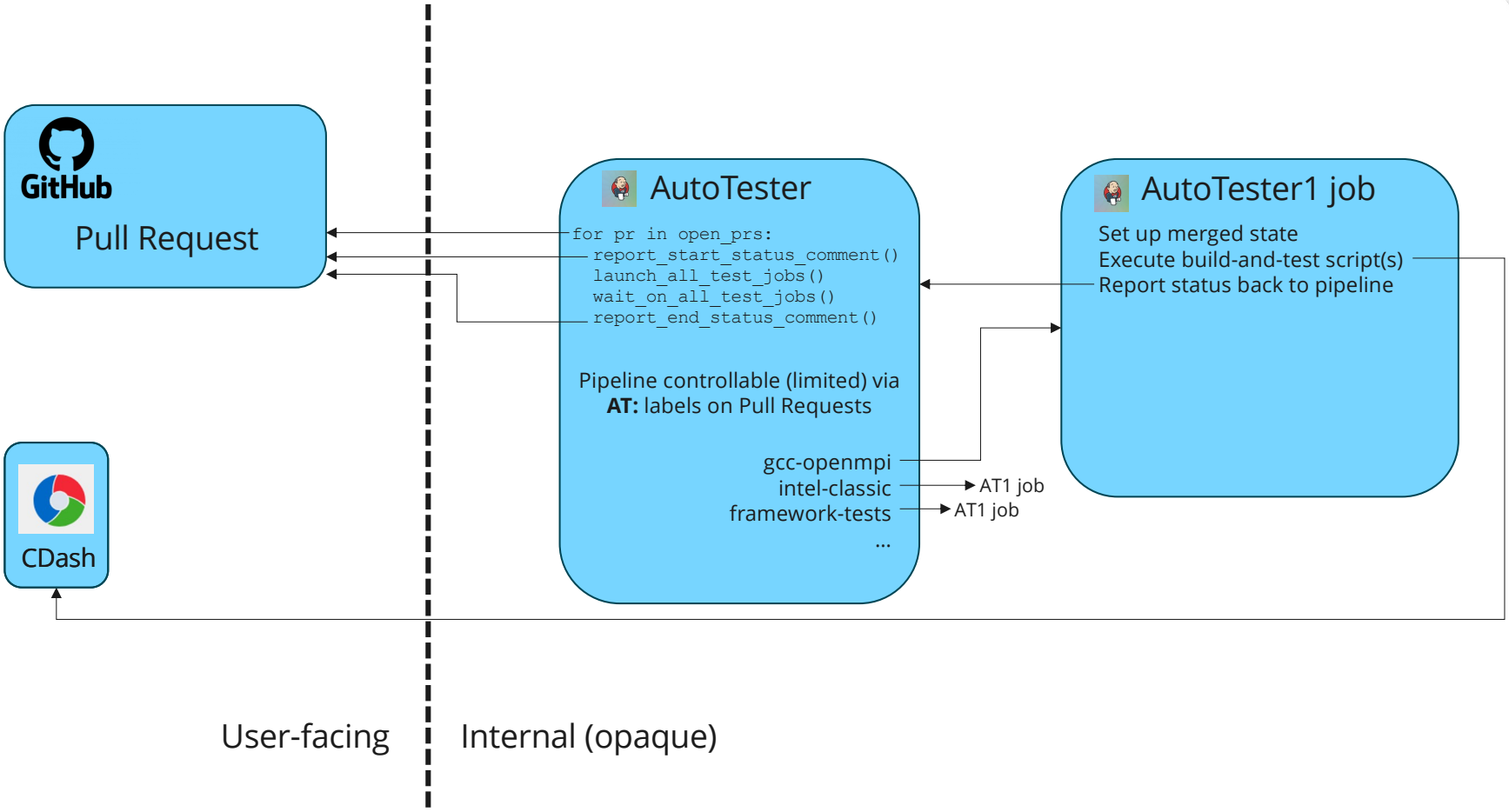
Special thanks to Anderson Chauphan, Joe Frye, and Justin LaPre for their contributions to the team and to this presentation.

HOW TRILINOS CI TESTING WORKS

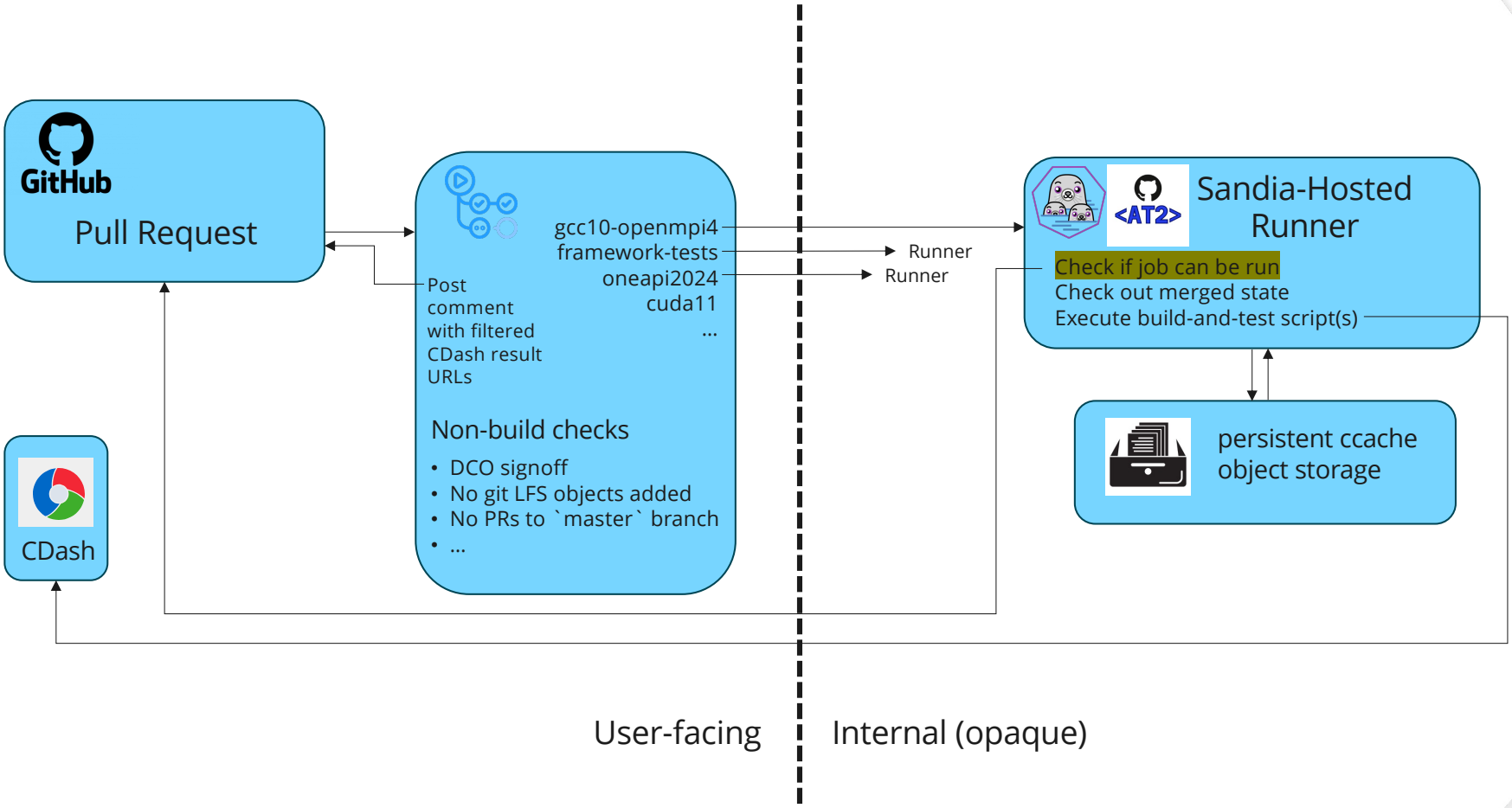
CI TESTING

- Trilinos uses a modified version of the “Gitflow” workflow with a `develop` and `master` (main) branch, where developers merge changes to `develop`, and the summation of those changes is merged to `master` on an approximately-weekly cadence.
- As part of this workflow, Trilinos has CI-style checks running through several different automation frameworks to ensure code quality
  - Required
    - A fairly comprehensive set of package builds and tests succeed on multiple toolchains
    - No commits contain Git LFS objects (causes mirroring complexity for internal repositories)
    - All commits have a DCO signoff
    - Commits which touch packages that have opted-in to clang-tidy conform to the standard
    - MPI_COMM_WORLD is not used in any modified lines of code
  - Non-required
    - CodeQL static analysis
    - Upcoming/experimental builds

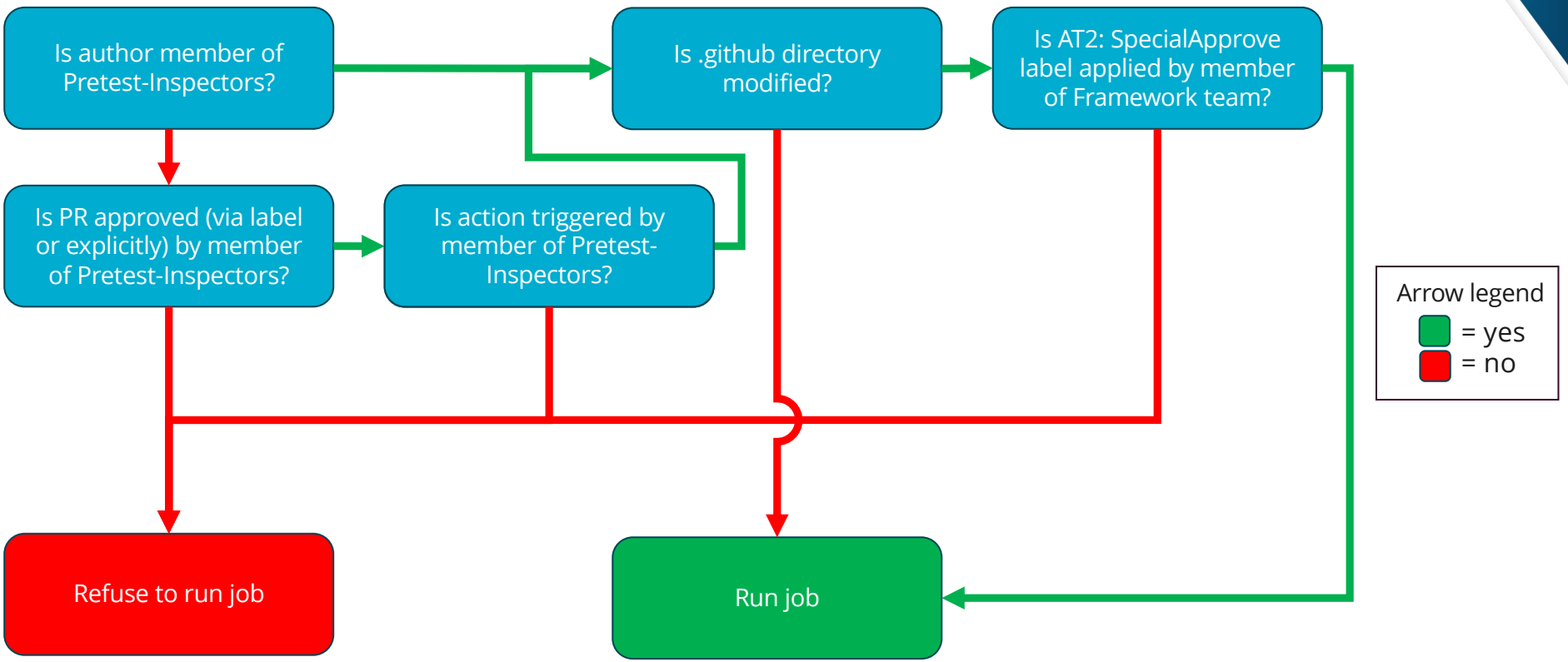
HOW DOES TRILINOS CI V1 WORK?



HOW DOES TRILINOS CI V2 WORK?

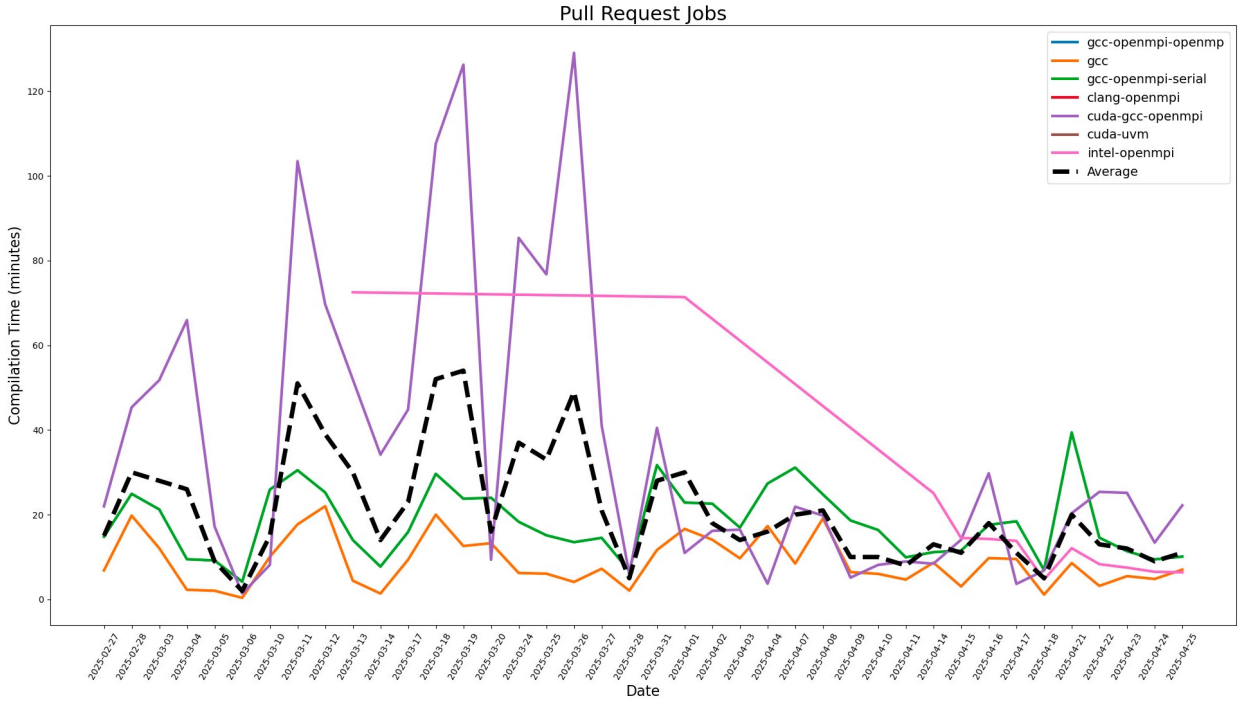


TRILINOS AT2 RUNNER DECISION TREE

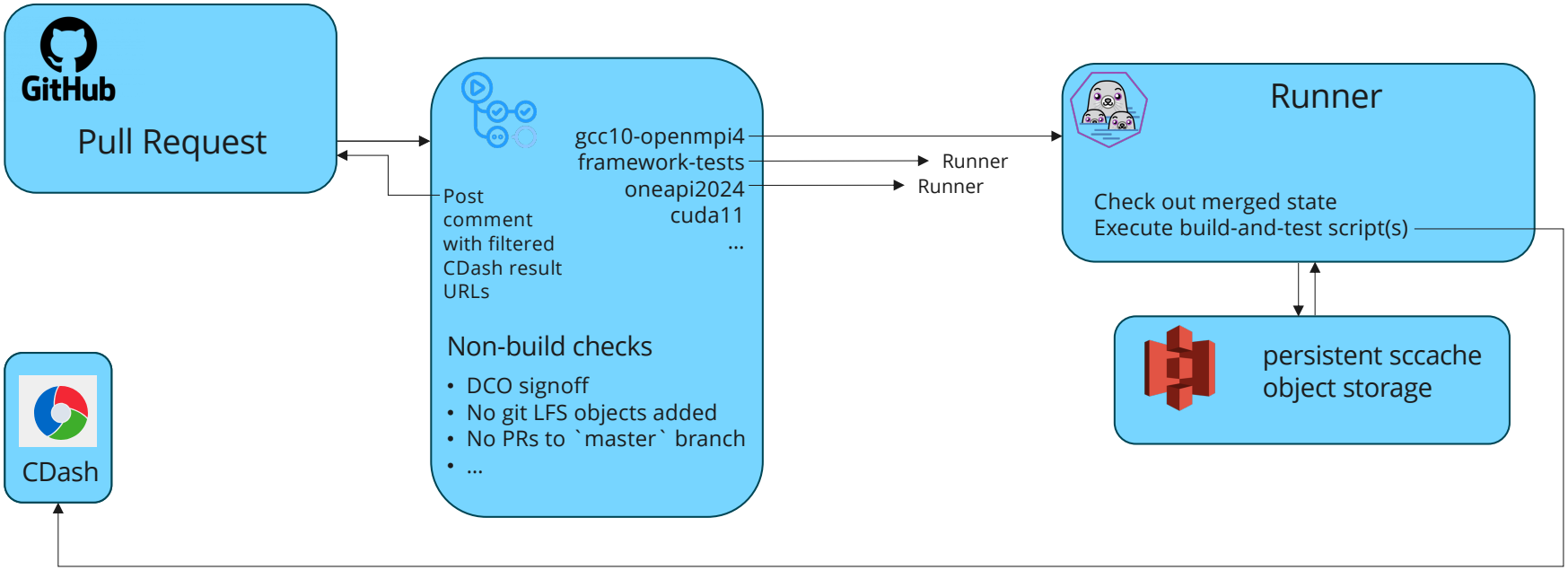


A NOTE ON BUILD SPEED (WHY CCACHE MATTERS FOR TRILINOS)

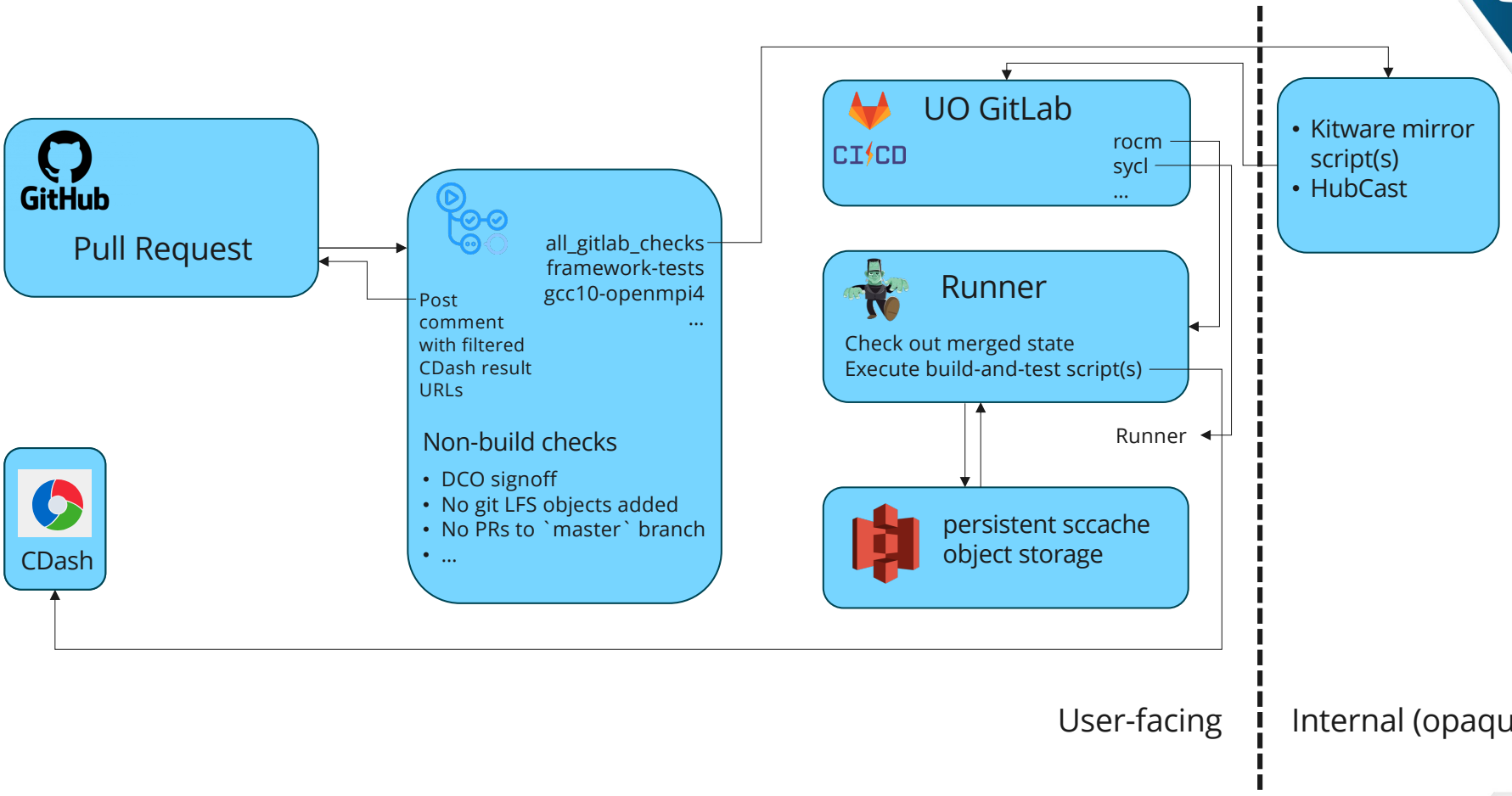
- Building Trilinos packages from scratch is fairly compute-intensive, and when running in a CI context, many duplicate compilations are performed run-to-run
- Using a per-node filesystem compiler cache (ccache) is giving a current direct hit rate of 88-91% depending on toolchain



HOW DOES TRILINOS CI V2 WORK WITHOUT SANDIA RESOURCES?



HOW COULD TRILINOS CI V2 WORK WITH UO/CASS RESOURCES?



THE FUTURE OF TRILINOS CI

- Encapsulating build/run environments as containers is incredible beneficial as compared to large toolchain and TPL stack deployments on internal machines
  - Allows customers to reproduce exact CI testing environment
  - Trilinos can have a CI testing environment separate from Sandia hardware if necessary, but can also utilize Sandia hardware in a relatively seamless manner
- Accelerated build and tests
  - Exploring per-package binary caching with Spack approach (later in this presentation)
  - Continue to use ccache on hardware supporting mounted filesystems, but explore sccache for external resources
- Make Trilinos configurations as easy to understand and interact with as possible

MAKING TRILINOS EASIER TO BUILD

MINIMIZING CONFIGURATION OPTIONS

"Trilinos is too difficult to build" – every surveyed user

- Most surveyed users tend to get a set of configuration options that work for them, then avoid changing any unless necessary

- Trilinos should strive to minimize the number of configuration options that must be passed, and ensure that sensible defaults are chosen to adjust the default configuration for the majority of users

- This proliferation of configure options can be seen in the CI configurations

MINIMIZING CONFIGURATION OPTIONS

Old CI example

- ~250 total config options passed
- 145 related to TPL related
- 15 package disables*
- 40 test disables
- 50 global/package-level configuration

New CI example

- ~200 total config options passed
- 115 related to TPL related
- 15 package disables*
- 20 test disables
- 50 global/package-level configuration

Are they all necessary?

MINIMIZING CONFIGURATION OPTIONS

Old CI example

- ~250 total config options passed
- 145 related to TPL related
- 15 package disables*
- 40 test disables
- 50 global/package-level configuration

New CI example

- ~200 total config options passed
- 115 related to TPL related
- 15 package disables*
- 20 test disables
- 50 global/package-level configuration

Are they all necessary?

MINIMIZING CONFIGURATION OPTIONS – THIRD-PARTY LIBRARIES

- TPL options are system-dependent
  - If a user has only one installation of LAPACK, `-DTPL_ENABLE_LAPACK` is generally sufficient
  - If a user has six installations of LAPACK, additional specificity may be needed for robustness (example of *necessary configure options*)
- New CI testing containers contain environments set up to Just Work with the single enable option
- At a minimum, must have the correct enables and disables
  - Exploring setting more sensible defaults
- Moving towards ”standard” `find_package()` as CMake package ecosystem matures will help decrease this complexity

MINIMIZING CONFIGURATION OPTIONS – PACKAGE DISABLES

- Package disables are entirely due to deprecated packages, and will be eliminated at the end of this year

MINIMIZING CONFIGURATION OPTIONS – TEST DISABLES

- Disabling Trilinos tests is *generally* due to a test that exhibits unstable behavior when running in CI testing
- General position is that while unstable tests are problematic for CI testing, they may still provide developers (or users) helpful information, so disable as part of CI instead of deleting

MINIMIZING CONFIGURATION OPTIONS – GLOBAL CONFIGURATION

- Largest spot for individual improvement within Trilinos packages
- Set defaults intelligently within multiple packages
  - Eliminate the case where one must pass options per-package:
    - `set(TPL_ENABLE_CUDA ON CACHE BOOL "from .ini configuration”)`
    - `set(Kokkos_ENABLE_CUDA ON CACHE BOOL "from .ini configuration")`
    - `set(Phalanx_KOKKOS_DEVICE_TYPE CUDA CACHE STRING "from .ini configuration")`
- Eliminate explicit setting of default options
  - Test the defaults!

SPACK

- Trilinos currently has ~80 Spack variants
- ~30 of them related to enabling non-deprecated packages
  - Casual or first-time users may not understand this, and see Trilinos as a monolith
  - How to represent Trilinos packages more understandably to users, while preserving the level of granularity of other configuration options?

- Proposed solution: Model Trilinos packages as individual Spack packages
  - `trilinos+sacado+muelu+cuda --> sacado+cuda muelu+cuda`
  - Already have a limited example of this with Kokkos and KokkosKernels (though only external “snapshotted” packages)

- Leverage knowledge and approach used for SNL Sierra code, see Phil Sakievich’s presentation at 4:40PM on Thursday

CONCLUSION

- New CI infrastructure leverages tooling familiar to much of the open-source community (GitHub Actions, containerized development environments, podman) and will help make the Trilinos development/contribution process more accessible

- Emphasis on reducing the number of configuration options *required* to produce a build of Trilinos that is desirable for most customers (to the degree possible)

- The Spack/Trilinos interaction is being worked to improve Trilinos user experience when using the de-facto standard build orchestration tool for HPCs and scientific software