www.ideas-productivity.org

IDEAS-EXAALT Collaboration: Adopting Continuous Integration for Long-Timescale Materials Simulation

Richard J. Zamora[1], Christoph Junghans[2], and David Moulton[2]
[1]Argonne National Laboratory, [2]Los Alamos National Laboratory

EXASCALE COMPUTING PROJECT

Overview

In order to leverage extreme scales efficiently, many modern applications are composed of a collection of distinct packages and libraries. For these projects, the necessary implementation of sustainable software practices requires developers to navigate multiple code bases. One promising answer to this challenge is the Productivity and Sustainability Improvement Planning (PSIP) methodology being developed by the Interoperable Design of Extreme-scale Application Software (IDEAS) project. In this work, we highlight a recent PSIP-based effort to implement an end-to-end continuous-integration pipeline within the EXAALT application-project software repository.

This work is highlighted in a September 2018 **Better Scientific Software (BSSw)** blog post: https://bssw.io/

The **EXAALT** Simulation Framework

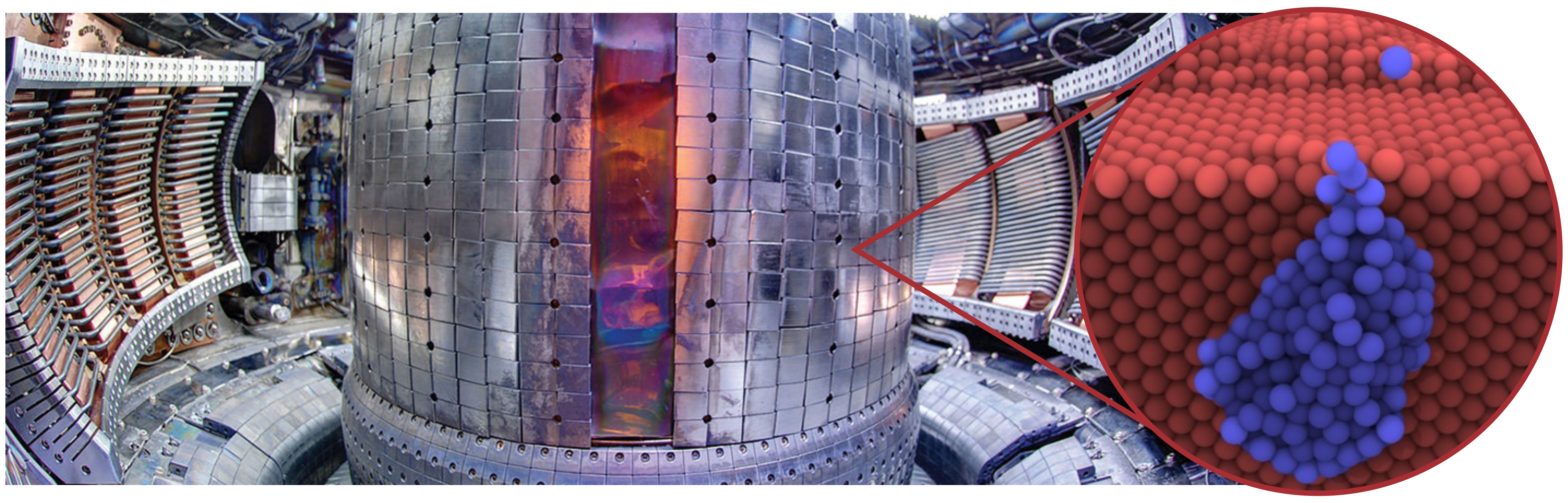


Figure 1. One application of EXAALT is modeling the surface of a fusion reactor (shown above is the interior of a tokamak at MIT, photograph by Chris Bolin, wikimedia commons). Simulation image credit: Luis Sandoval.

The **Exascale Atomistic Capability for Accuracy, Length and Time** (**EXAALT**) is a materials simulation framework allowing users to access the most appropriate combination of accuracy, length, and time for the problem at hand; trading the costs of various forms of parallelism. EXAALT is actually a collection of three sub-projects (ParSplice, LAMMPS, and LATTE), each with its own development processes and dependencies.

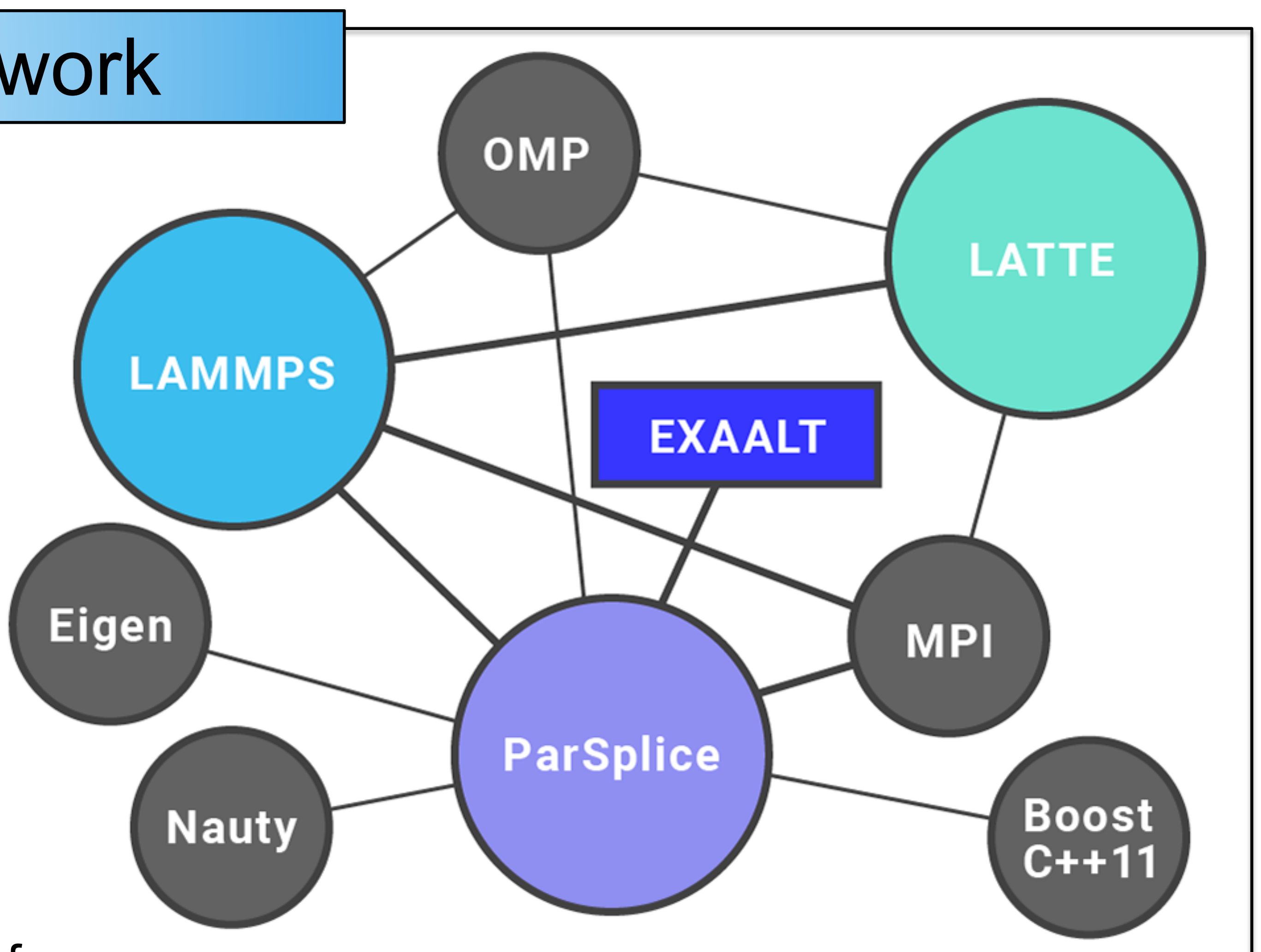


Figure 2. Illustration of the EXAALT framework. The three main software components (LAMMPS, LATTE, and ParSplice) are represented as colored circles, while other libraries are represented as grey circles. Lines (graph edges) depict dependencies between the various software components.

Continuous Integration (**CI**) in EXAALT

Continuous Integration (CI) is a software development process that relies on the automated compilation and testing of all new features to detect bugs early. In order to implement a preliminary CI pipeline in EXAALT, we leverage the following tools:

- CMake: Used to manage the compilation of EXAALT and then to execute functionality tests (using CTest) for each build.
- Boost: Used to implement and organize functionality tests (integration, regression, and unit) inside CTest.
- GitLab CI: Used to automatically build and test the software framework (using CMake) to validate new repository commits.
- Docker: Used to generate standard system images (with library dependencies) for use in GitLab CI.

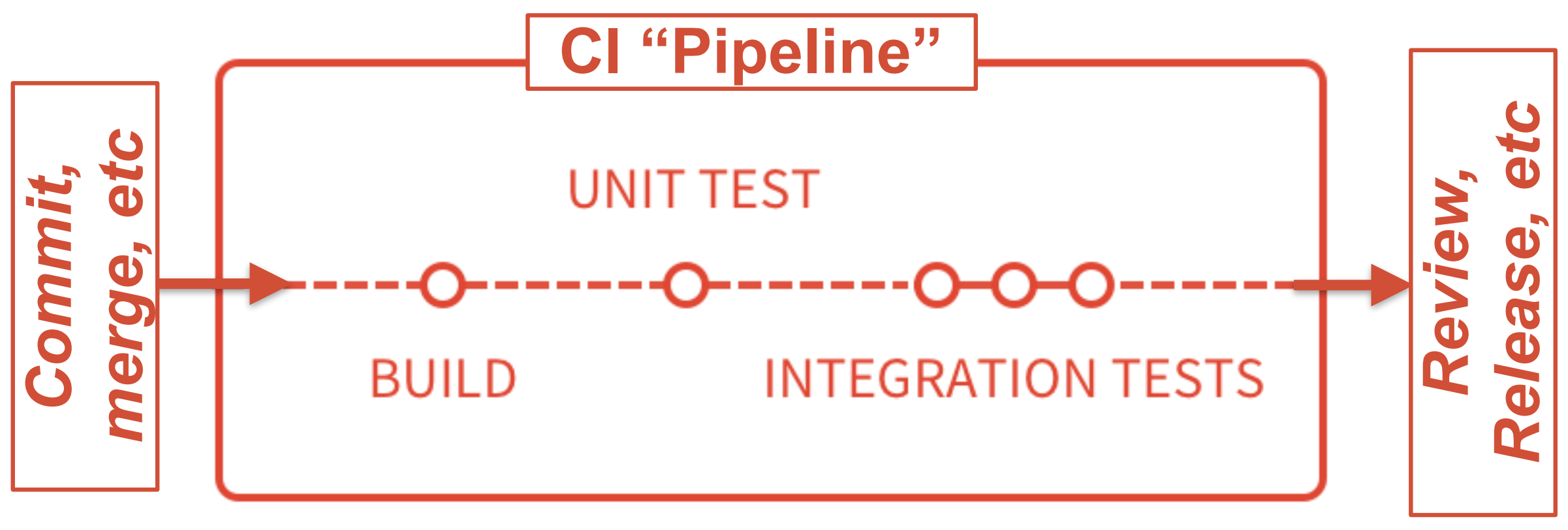


Figure 3. Illustration of a continuous integration pipeline. For EXAALT, the preliminary pipeline was implemented using GitLab CI (along with other complimentary tools).

Applying the **PSIP** Workflow

Productivity and Sustainability Improvement Planning (PSIP) promotes the clear factoring of new software processes and capabilities **into a manageable number of critical steps with simple completion criteria.** These steps are used to track the overall progress of the project toward some *target* status (the long-term goal).

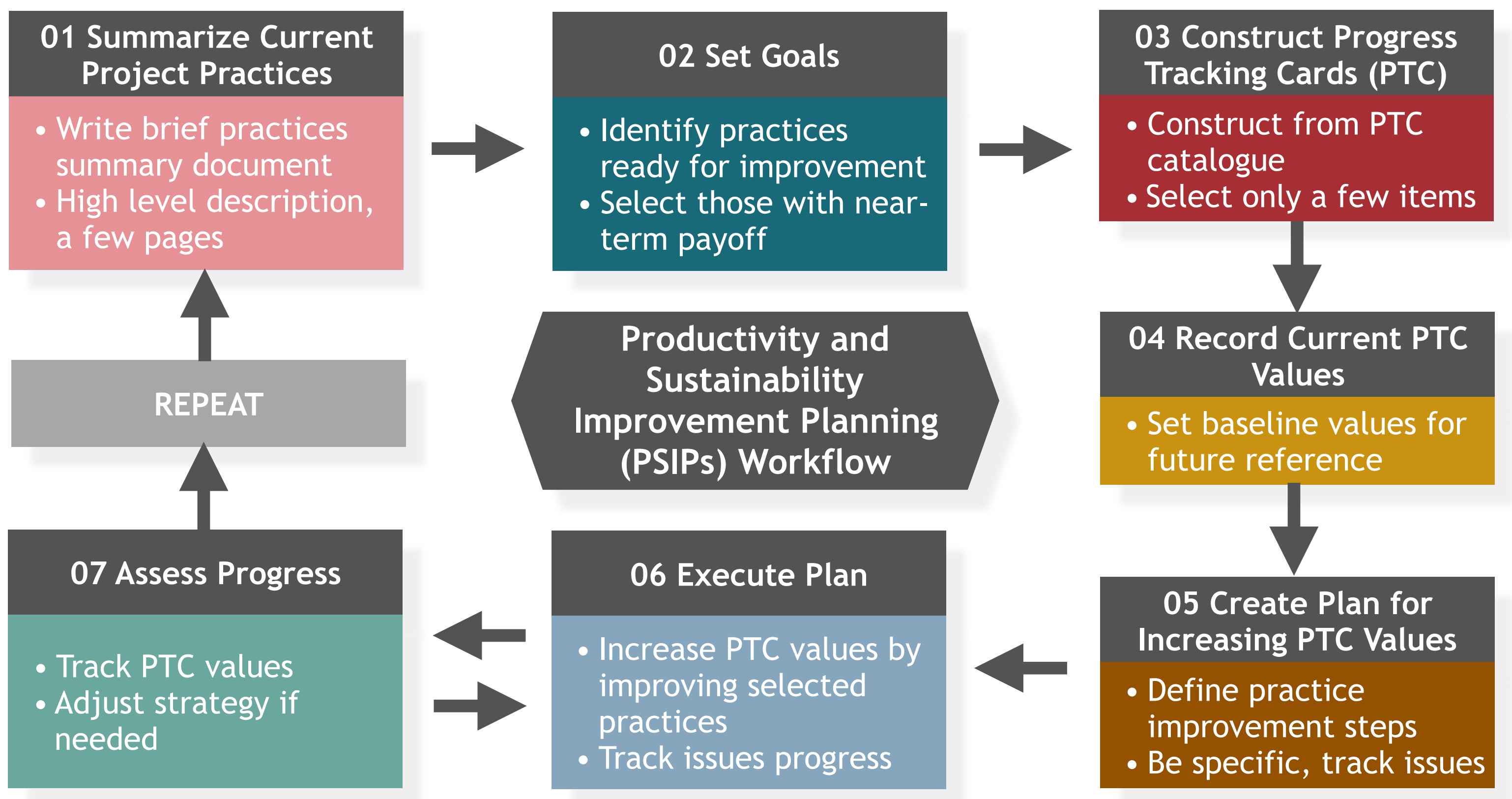
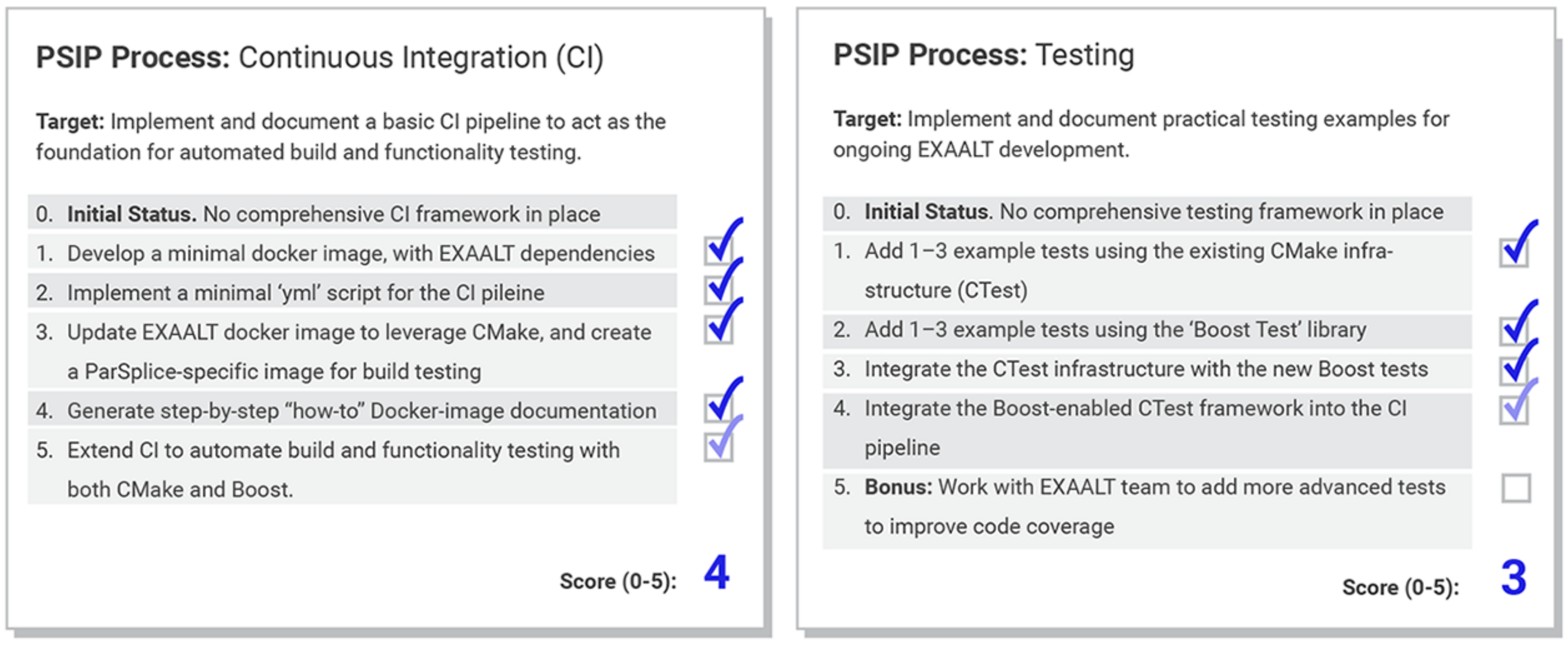

01 Summarize Current Project Practices
• Write brief practices summary document
• High level description, a few pages
02 Set Goals
• Identify practices ready for improvement
• Select those with near-term payoff
03 Construct Progress Tracking Cards (PTC)
• Construct from PTC catalogue
• Select only a few items
04 Record Current PTC Values
• Set baseline values for future reference
05 Create Plan for Increasing PTC Values
• Define practice improvement steps
• Be specific, track issues
06 Execute Plan
• Increase PTC values by improving selected practices
• Track issues progress
07 Assess Progress
• Track PTC values
• Adjust strategy if needed
REPEAT
Productivity and Sustainability Improvement Planning (PSIPs) Workflow

Figure 4. Illustration of the general PSIP workflow.



PSIP Process: Continuous Integration (CI)

Target: Implement and document a basic CI pipeline to act as the foundation for automated build and functionality testing.

0. **Initial Status.** No comprehensive CI framework in place
1. Develop a minimal docker image, with EXAALT dependencies ✓
2. Implement a minimal 'yml' script for the CI pileine ✓
3. Update EXAALT docker image to leverage CMake, and create a ParSplice-specific image for build testing ✓
4. Generate step-by-step "how-to" Docker-image documentation ✓
5. Extend CI to automate build and functionality testing with both CMake and Boost. ✓

Score (0-5): 4

PSIP Process: Testing

Target: Implement and document practical testing examples for ongoing EXAALT development.

0. **Initial Status.** No comprehensive testing framework in place
1. Add 1–3 example tests using the existing CMake infra-structure (CTest) ✓
2. Add 1–3 example tests using the 'Boost Test' library ✓
3. Integrate the CTest infrastructure with the new Boost tests ✓
4. Integrate the Boost-enabled CTest framework into the CI pipeline ✓
5. **Bonus:** Work with EXAALT team to add more advanced tests to improve code coverage ☐

Score (0-5): 3

Figure 5. Summarized versions of the PSIP process cards used for the EXAALT-IDEAS collaboration. The specific scores correspond to the state of the project in mid-July 2018: Boost-enabled tests have been added to the CMake build system, and the existing Gitlab CI pipeline needs to be modified to leverage the current CMake/CTest capabilities. Lightly-shaded check marks correspond to the PTC steps that were "in-progress" at the time.

SIAM CSE19, February 26, 2019

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.