LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-854075

Level-2 Milestone 8551: Development Environment for El Capitan

E. Gonsiorowski, J. Gyllenhaal, W. Futral, M. LeGendre, R. Pankajakshan, N. Sly, G. Lee, C. Chambreau

September 11, 2023

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL-XX-NNNNNN

Lawrence Livermore National Laboratory

Level-2 Milestone 8551: Development Environment for El Capitan

Prepared by: Elsa Gonsiorowski, John Gyllenhaal, W. Scott Futral, Matthew Legendre, Ramesh Pankajakshan, Nicholas Sly, Gregory L. Lee, Chris Chambreau

August 31, 2023

Disclaimer

Table of Contents

1 Executive Summary

This report describes the ASC L2 Milestone #8551 "Development Environment for El Capitan" which documents the deployment and use of the development environment for the El Capitan Early Access Systems (EAS-3). The milestone reads:

> This effort will build a development environment for the El Capitan system. The development environment will include compilers, tools, programming model runtimes, build tools, and other software infrastructure that will be needed to build applications on El Capitan. This effort will also develop documentation and training opportunities.

Since the El Capitan system will not be ready at this milestone's due date, the development environment will be built and prototyped on the EAS-3 systems. It will be based around the TCE2 software stack, which will allow it to share much (though not all) of the development environment found on CTS-2 TOSS4 systems.

The milestone completion criteria states:

> A portable development environment will be provided on the EAS-3 systems, and an evaluation of the environment will be performed by an ASC code team.

This report describes the successful effort undertaken by the LLNL Livermore Computing (LC) group to create and document the development environment for the EAS-3 platforms. The development environment includes:

- the user environment (as deployed through modules),
- compilers,
- MPI and OpenMP runtimes, and
- software tools (i.e., performance analysis tools, debugging tools, and correctness checkers).

The development environment described in this report has been successfully used by several ASC code teams. The LC team has worked closely with code team users and vendor personnel to identify issues, document workarounds, and deploy software updates. The development environment has been built such that it can be easily deployed to the future El Capitan systems.

In support of this milestone, 5 letters of support from ASC code teams are included (Appendix A). A snapshot of the user documentation wiki site for the EAS-3 systems is also appended (Appendix B). User and vendor educational presentations are included in Appendix C.

2 Introduction

LLNL and Livermore Computing (LC) have a reliable history in siting successful Advance Technology Systems (ATS), with Sierra as an active example. With the forthcoming El Capitan systems, we have the opportunity to improve upon our user environment philosophy and

implementation. For this next generation of systems, LC has capitalized on the chance to meet additional design goals to improve the overall experience of users on GPU-based ATS systems.

With this opportunity, we now provide three distinct environments for our Tri-Lab code developers:

1. A vendor-provided, module-driven environment which allows for portability across HPC facilities;
2. A Spack-friendly, module-less environment;
3. An established LC-style environment, allowing portability from previous ATS (e.g., Sierra).

To successfully develop these user environments, intensive vendor and user engagements have taken place. From a vendor's perspective, it is essential that they understand the expectations of the ASC code teams. For users, clear communication of available technologies and features is necessary. As for any ATS deployment, success on the El Capitan platform will come from the interaction between users, facilities, and vendors.

In addition to improving the experience for code developers, LC has revised the user environment deployment methodology. The improved deployment methodology is based on the Spack project and allows for a more powerful and flexible user environment. This approach leverages Spack’s large open source community that has arisen over the past decade. Spack is a foundational technology which can enable a common development environment across the Tri-Labs.

In addition to developing and deploying the user environment overall, ongoing efforts to improve the compilers, parallel programming paradigms (i.e., MPI and OpenMP), and tools for debugging, correctness checking, and performance analysis have taken place.

This milestone report documents the efforts made to develop and deploy the user environment for the El Capitan Early Access Systems (EAS-3), namely the Tioga, RZVernal, and Tenaya systems at LLNL. The report is organized as follows. We start by describing our user environment philosophy and analyzing the benefits for each user environment approach (Section 3). Section 4 and 5 discuss vendor and user engagement efforts, respectively. We detail the user environment deployment methodology in Section 6. Section 7 reports on efforts made for specific tools (e.g., compilers) followed by a brief description of ongoing efforts (Section 8) and conclusions (Section 9).

3 User Environment Philosophy

The Cray user environment has a long and robust history where it has been successfully used at many DOE facilities and for some (non-LLNL) Tri-Lab ATS deployments. Indeed, many supercomputer facilities have successfully relied on solely the Cray module-based user environment as provided by the vendor. However, Cray user environment as it existed at the start of the El Capitan contract was not well suited for many of LLNL's ASC codes' build systems. Nor was it well suited for use with Spack, which is continuing to see wide-spread adoption across the DOE and the world. With El Capitan, LLNL has the opportunity to work with HPE/Cray to develop the improvements needed by our code teams.

With this is mind, LLNL is providing three distinct user environments for our Tri-Lab code developers:

1. **Cray's Module-Sensitive Environment**: This vendor-provided, module-driven environment allows for portability across HPC facilities. This is the default environment on EAS-3.
2. **HPE/Cray's New Module-Less Environment**: This is a Spack-friendly, module-less environment that has full buy-in and support from our vendor partners. It has required extensive co-design between LLNL, HPE, and AMD.
3. **LC's Magic Environment**: This is the traditional LC-style environment which allows portability from previous ATS (e.g., Sierra). For the El Capitan deployment, LLNL has taken the opportunity to reengineer the underlying deployment methodology.

Each of these environments and the philosophies behind them are detailed in the sub-sections below.

3.1 Cray's Module-Sensitive Environment

For decades, the Cray Programming Environment (Cray PE) has been based around modules. The philosophy behind this module-sensitive user environment is that a simple driver (cc for C, CC for C++, and ftn for Fortran) automatically picks a compiler and adds all the compiler options, flags, includes, and libraries required by the application based on the Cray PE modules loaded. This environment allows users to specify the compiler vendor and version to use (e.g., cce/16.0.0, amd/5.6.0, gcc-native/10.3, etc.), the MPI vendor and version to use (e.g., cray-mpich/8.1.26), the network fabric to use (e.g, craype-network-ofi and libfabric/2.0), the CPU to optimize for (e.g., x86-trento, x86-genoa), the GPU to target and enable gpu-aware MPI for (e.g., amd-gfx90a, amd-gfx940), specialized math libraries and versions to load (e.g., cray-libsci/23.05.1.4), and more.

There are several benefits of this module-driven approach. First, users have very short compiler invocations, such as 'CC -O3 mpi_app.cc', as all additional flags are handled by the driver. Additionally, it is trivial to switch between compilers, MPIs, etc., while using the same makefile; users simply switch modules loaded. Many users, particularly those with simple to build codes, really appreciate the power and flexibility that this approach enables.

Unfortunately, for Spack-based and for complex build environments that ASC codes often have, the module-sensitive Cray environment makes it extremely difficult to get reproducible builds, both for the same code developer and for different code developers on the same code team. Slightly different modules loaded, or even modules loaded in a different order, can cause different compiler options to be used resulting in differences in the end application. In addition, subshells in complex Makefiles can cause the loaded modules to change unexpectedly mid-build. Many hacky and high-maintenance approaches have been developed to attempt to work around the difficulties the module-sensitive Cray environment can create.

3.2 HPE/Cray's New Module-less Environment

Through the El Capitan contract, LLNL has driven an extensive evolution of the default Cray user environment. This evolution has been implemented by and has the full support of HPE/Cray. We anticipate that future HPE/Cray system deployments, even those at other

facilities, will include these improvements increasing the portability of ASC codes. This new, module-less user environment has been explicitly designed for Spack support.

This user environment is the result of collaboration and co-design with HPE and AMD. The activities are centered around the CORAL2 working groups, particularly the packaging WG, the compiler WG, and the messaging WG. Both the Cray and ROCM user environments, as deployed on El Capitan **and at other facilities**, now fully support module-less operation. This environment is well-suited for use with Spack and for building ASC codes.

In this module-less environment, only the path to compiler called specifies the version to be used (e.g., /opt/cray/pe/cce/16.0.0/bin/crayCC) and the compiler behavior does not change with what modules are loaded. Similarly, traditional MPI wrappers are now provided where only the path to the wrapper (e.g., /opt/cray/pe/mpich/8.1.26/ofi/crayclang/16.0/bin/mpicxx) and the expected environment variables, (e.g., MPICH_CXX), affect what compiler and MPI is used. All other compiler options, include files, and libraries must be explicitly specified on the compile and link lines, leading to reproducible builds with explicit options. This enables Spack to function as it was designed. In the past, Spack has had to rely on Cray-specific workarounds. Now, it can treat Cray machines like any other Linux system.

3.3 LC's Magic Environment

Historically, the LC user environment has been an essential part of the reproducible builds for LLNL ASC codes. As such, LLNL is committed to deploying a similar environment on any platform it hosts. This LC-style user environment is currently provided on the Sierra ATS and was in place for previous IBM ATS generations. Thus, code development teams familiar with these systems will have a straight-forward path to porting to EAS-3 and El Capitan.

While not previously explicit, in the course of developing the El Capitan user environment LLNL has had the opportunity to name and formally define the principles underpinning its user environment; this is the LC Magic Environment.

The key feature of the LC magic environment is driven by the ASC program, that is: the behavior of a previously built application does not change with system updates. In this way, ASC code teams require 'static' builds, despite utilizing shared libraries. In the past, random changes in application behavior due to system updates has ruined year-long UQ or sensitivity studies. With the LC magic user environment, ASC code teams create "golden" executables which have been extensively verified. These executables are then used throughout the life a machine. In addition, the LC magic environment allows builds from years ago to be recreated today. Thus enabling developers to search through years of code changes to identify a root cause that unexpectedly affected performance or correctness.

This user environment approach was developed in the early 2000s to facilitate and streamline the development of huge ASC codes, with the goal of eliminating all the common build gotchas that had continually tripped up many experienced code developers in the ASC program. Literally several person-years of effort was wasted early on in ASC code development due to tracking down application issues caused by unexpected or unnoticed changes in the user environment during build- or run-time. The key idea behind the LC "magic" versions of compilers and MPI wrappers was that the full path specified behavior, and that no environment variable (including MPICH_CXX) would affect what the compiler did. In addition, the "magic" versions would specify RPATHs and libraries such the behavior of the generated executables would be

independent of practically all commonly set environment variables, including LD_LIBRARY_PATH. (LD_PRELOAD does exist as a workaround, but many users end up causing themselves unintended problems.)

The LC magic environment has allowed *all* code developers to be able to reproduce the same executable and for *all* users of the code to get the same behavior no matter who was running the code. This environment strategy has returned massive dividends for the productivity of our ASC code developers over the years, again and again. However, these benefits used to only occur on LC supercomputers until Spack was developed (at LLNL). Spack provides a more portable approach to providing these same benefits to code developers in addition to working on non-LLNL machines. Once all LLNL ASC codes start building with Spack (or something similar), it is hoped that the LC magic environment will no longer be necessary in future ATS deployments.

The deployment methodology for the LC magic environment has continuously evolved over the decades. Each new platform and generation of systems is another opportunity to improve and even re-design certain aspects. For El Capitan and RHEL-8-based systems, LLNL has re-implemented the way in which the user environment is deployed. The new deployment methodology builds as much as currently practical with Spack's new environment features. Our goal is to eventually being able to deploy identical environments (or key pieces of El Capitan's environment) at other sites.

This feeds into the long term goal of having common compute environments across the Tri-Labs. Spack continues to mature, and this vision is still being explored by many supercomputer sites (with informative recent research publications being a result). Our experiences in using Spack technology to deploy the EAS-3 user environment has driven significant Spack development to both simplify and harden the process. We envision a future where the entire LC magic user environment could be generated from a production Spack release.

4 Vendor Engagement

In our experience, successful ATS deployments require dedicated time and effort for vendor and facility collaboration and co-design. Software design based *only* on short summaries of what is to be done, even though the summary appears to describe exactly what is desired, is almost guaranteed to lead to major surprises at delivery time. Success requires frequent, in depth discussions between Tri-Lab technical staff and the actual team doing the work for the vendor. Both teams continually deliberate on designs, design tradeoffs, critical features, usage models, and acceptable limitations.

As a result, we design our vendor ATS contracts to require this interaction (usually via significant money tied to NRE deliverables) and allocate significant lab staff time to drive these critical collaborations. This occurs from the time the contract is signed until the machine and promised software is delivered and accepted. The requirement for deep interaction often seems initially awkward and onerous to new vendors, but over time they see that these efforts regularly prevent significant amounts of unnecessary work and allow them to prioritize the actual needs of the Tri-Labs for their deliverables. Many vendors have come to embrace and value these deep collaborations.

An essential~~The~~ medium for collaborations ~~are~~ is the working groups (WG). While there are ~~several~~several- working groups for the El Capitan contract, a subset are particularly involved in user environment topics, as detailed below. Working groups interact regularly through scheduled telecons and continuous email communications. WG's~~all~~ come together for dedicated, face-to-face interactions during quarterly meetings. In addition to these collaborations between facilities staff and vendor teams, the El Capitan Center of Excellence (COE) is an avenue for direct engagement for vendor technical staff and code team developers. Any issues identified relating to the user environment are funneled through COE channels, and weekly "bug-scrubs" review outstanding items.

4.1 Face-to-Face Meetings

Face-to-face interactions are critical for building common understanding and trust between previously independent teams. Particularly for detail-oriented, technical teams, it is important that both sides gain perspective on where the others are coming from. These face-to-face interactions give folks a chance to not only address deep technical issues, but also a chance to mingle, socialize, and build team morale.

4.1.1 Quarterly Meetings

Four times a year, the El Capitan vendor partners and the appropriate DOE labs have three- ~~-~~day Quarterly meetings either at a vendor site (HPE, AMD) or at a lab site (LLNL, ORNL). During these meetings, both managers and technical staff working on El Capitan get together to synchronize on the current state of hardware and software. These meetings ensure all parties are on the same page about current status and future plans. Selected working groups have half- or full-day breakout meetings during these quarterly gatherings.

4.1.2 March 2023 User Environment Deep Dive

In addition to regularly scheduled Quarterly Meetings, ad hoc face-to-face meetings also occur. One such meeting happened in March 2023. At this time, LLNL sent ~~d~~ six key technical staff to HPE for a two day deep dive. This deep dive focused on the LC M~~m~~agic environment and detailed ~~ing~~ the requirements of the ASC code development teams.

LLNL has not worked with HPE/Cray on an ATS deployment in recent years. Thus, this meeting was a critical opportunity for HPE development environment and testing personnel to familiarize themselves (through hands-on-experience) with LLNL environments. It is vital that the El Capitan vendors understand both the LLNL OS install process and the LC Magic user environment. ~~Through this understanding, t~~The vendor teams can apply this understanding to better focus ~~appropriate~~ development and testing of ~~t~~ their software locally before delivering to LLNL.

The March 16th, 2023 development environment presentation titled "Explaining LLNL's '/usr/tce' magic to HPE: The whys. how, and future plans for El Cap" is provided in Appendix C.

At the end of this meeting in March, many members of HPE staff thanked LLNL for this opportunity. They had previously found the LLNL user environment philosophy to be counter intuitive. Finally, support requests made sense to them, and they could understand the reasoning behind many of our user environment requests. This has directly resulted in improvements to the

HPE/Cray module-less user environment that will be included on *all* future HPE system deployments. ASC code teams will have a measurably easier time in porting to such systems.

4.2 Working Groups

Working groups are established to ensure regular contact between vendor and facility technical teams. Often, their interactions are focused on NRE deliverables and acceptance milestones.

4.2.1 Compiler WG

The ATS compiler working groups have been a key part of ATS deployments for the last two decades. For El Capitan, the Compiler WG started in August 2019. The WG meets at least once every month, in addition to the six hours of dedicated time during the Quarterly meetings. The primary focus has been on OpenMP GPU language standard support (which features to implement first, testing, optimization strategies, etc.), HIP support and critical bugs, and GPU performance improvement and correctness issues. These meetings are focused on technical discussion for how to address various issues, often allowing key Tri-Lab use cases to be explored. Through these interactions, vendor technical teams gain perspective on the needs of the Tri-Lab code teams, leading to better long-term solutions. In addition, key OpenMP standard committee members attend all meetings, allowing guidance to solutions that will work better with proposed future OpenMP standards.

4.2.2 Messaging WG

The ATS messaging working is another key part of many ATS deployments. The Messaging WG for El Capitan specifically started in August 2019 and focuses on network hardware and the network software stack. This group meets for two hours once a month and typically holds a six-hour meeting during the Quarterly meetings. These interactions focus on the approaches and known issues with performance and correctness for MPI, including GPU-aware MPI.

4.2.3 Packaging WG

The Packaging WG is new for the El Capitan deployment. It was created in December 2020 once the need for collaboration on the overall user environment deployment was identified. The Packaging WG focuses almost entirely on on how software is packaged appropriately for installation on the EAS-3 systems, and including how Spack is leveraged. The design and implementation of the new HPE/Cray module-less user environment was driven by this group. The Packaging WG has met for an hour roughly every two weeks since December 2020, plus six hours of meeting at most Quarterly meetings.

Some user environment improvements credited to the Packaging WG include:

- The design and implementation of Spack manifests for Cray software, which enable users to leveraging HPE provided software via Spack.
- Initial discussions around HPE/Cray providing 'standard' mpi wrappers such as mpicc, mpicxx, mpif90.

Through the Packaging WG, the vendor technical teams are able to ask 'why' and explore Tri-Lab user environment requests. These interactions allow for in-depth discussions, allowing the vendor teams to gain a greater understanding of the ASC code development worldview.

4.2.4 Tools WG

Both the Tools WG and Infrastructure WG have been a part of previous ATS deployments. Spun-up in December 2019, these groups focus on the development of both interfaces for tools and the tools themselves. These meetings focus on deep discussions around development timelines and priorities, ensuring that necessary tools are available to end users.

The Tools WG has spun up both NDA and non-NDA biweekly meetings to respectively discuss ongoing AMD and HPE tool efforts, and coordination with open-source tool developers. In addition, the Tools WG meets in-person at CORAL-2 Quarterly meeting. Through these meetings we have coordinated new profiling tool interfaces, correctness tool design, and resolved deployment issues.

4.2.5 Math Libraries and Machine Learning WG

The Math Libraries and Machine Learning WG focuses on the various high-performance CPU and GPU libraries users expect. These were begun in August 2019. The WG meets once a month to discuss NRE milestone reports and any technical issues that arise from the NRE work on math and ML libraries. Two sets of math libraries each with support for CPUs and GPUs are available to users on the EAS-3 systems.

4.3 Vendor Support for the Center of Excellence

The El Capitan Center of Excellence (COE) is composed of nearly 20 dedicated vendor staff (more than 10 full time employee equivalents) working alongside Tri-Lab code teams. These COE vendor staff are able to experience directly what our key ASC code teams need across the entire user environment. These staff are crucial to finding short term solutions to issues and driving internal vendor development in the appropriate direction.

4.3.1 Bug Scrubs

The COE vendor staff and LLNL user environment teams meet weekly to review all open user issues. From October 2021 to July 2023, over 425 user issues were reported, tracked, submitted to vendors, and worked-around. Around 200 issues have found resolution. These "bug scrub" meetings are the primary mechanism that LLNL uses to ensure user problems are solved and not dropped on the floor.

Separate bug scrubs involving more of the vendor staff (in non-COE related roles) are regularly carried out at each vendor site. Through these meetings, LLNL sets priorities for bug fixes, gets status on existing bugs, pushes on those that seem stalled, etc. These meeting have also spawned several deep dives with the technical teams. Deep dives are focused on solving specific bugs, particularly when bug reports were rejected or when the proposed solution appeared to be going in the wrong direction.

5 User Engagement

User engagement is critical for the success for any HPC platform, particular ATS deployments where many elements may be novel. For the El Capitan systems, user engagement has been focused on ensuring code teams are able to productively use the early access systems. Initially,

the El Capitan Center of Excellence (COE) was the main venue for engagement. Here, ASC developers covered under the appropriate NDAs were able to first access EAS-3 and begin the process of porting and optimizing their applications. As the systems' user environment became more stable, it was documented in the "El Cap EA Systems" wiki space. This documentation is actively updated and contains no NDA-sensitive information (though it is limited to only those with access to the systems). Efforts to harden the documentation and turn it into a tutorial presentation are ongoing. Tutorial materials will be made publicly available.

5.1 El Capitan Center of Excellence

The El Cap COE is engaging stakeholders across the Tri-Labs to ensure we "reach the summit" of El Capitan.

The most interactive venue for user engagement comes through the COE. With a dedicated LLNL liaison for each of the Tri-Labs, the COE acts as one of the means of communication for developers who are covered by the El Capitan NDA to interact with the vendors. These developers are able to gain early access to hardware and software that cannot be widely distributed. This accelerates the issue reporting, solution confirmation, and performance evaluation.

The COE engages developers through a number of in-person and virtual channels. Quarterly in-person hack-a-thons both solve problems and further educate key vendor technical personal on our needs. In addition, regular virtual meetings are held on user environment, hardware topics, and/or open discussion. These meetings provide timely training to Tri-Lab users on what is available in the user environment, including new performance tools and metrics to optimize codes. The COE also provides a central repository for user issues. For day-to-day interactions, an instant-message platform is also available.

5.1.1 Tri-Lab Hackathons

The COE hackathons are three-day, in-person events where vendor staff are available to interact directly with code developers. These events facilitate rapid iteration on issues and bugs and allow development teams to make huge progress and their coding efforts. Over the past year, hackathons have been held both at LLNL and Sandia, providing in-person opportunities for many ASC code teams.

The most recent hackathons where held:

- at LLNL, October 25-27, 2022,
- at Sandia, November 2-4, 2022,
- at LLNL, February 7-9, 2023,
- at Sandia, April 18-20, 2023,
- at LLNL, July 25-27, 2023.

The next hackathon is scheduled at Sandia in late October 2023.

5.1.2 Virtual Meetings

The virtual meetings are regularly scheduled to provide a forum for COE user training and user group discussion. Training have covered a diverse set of user environment and hardware topics, such as:

- Scheduling and concurrency on the MI300
- Roofline profiling on AMD hardware
- Introduction to register pressure in AMD compilers
- Bugs fixed in ROCM 5.5
- Resolving misconceptions about GPU-aware MPI
- What is Flux and why do I care? (Presentation slides included in Appendix C).

Meeting notes and recordings of formal presentations are archived in a COE web space.

5.1.3 Issue Tracking System

The COE issue tracking system a valuable mechanism for addressing user issues and engaging vendor / COE staff. The issue tracking system is run on the Jira platform and is housed in LLNL's restricted zone (RZ). The RZ allows for unclassified content which may be otherwise sensitive (e.g., export controlled). By hosting the issue tracker in the RZ, many users can directly share reproducers and/or code snippets. The issue tracker is restricted to NDA-covered individuals and includes most of the COE vendor staff.

It also helps users find existing workarounds for known issues since it is fully searchable.

The issue tracking system is the basis for the regular "bug-scrubs" (described in Section 4.3.1).

5.1.4 Instant Messaging Platform

The El Capitan COE has leveraged the NNSA-delopyed Mattermost application to provide an instant messaging service. Through Mattermost, users are able to quickly interact with both COE and LC staff to get timely help with their problems.

5.2 Documentation and Tutorial

The user environment for the EAS-3 platform has been documented as the “El Cap EA Systems” wiki space and is built on the Atlassian Confluence platform. This space is hosted in LLNL’s collaboration zone (CZ) and is open to any LC users who have been granted access to the Tioga, RZVernal, and Tenaya systems. See Appendix B for a snapshot of the documentation space from mid-August 2023.

First started in December 2022, this documentation space now covers a wide range of topics with a target audience raging from new LC users to experienced code developers. In particular, the documentation highlights the various modes for utilizing the user environment and provides both C++ and Fortran examples for programming to the AMD GPUs. In addition, email announcements are archived and work arounds to high-impact issues are documented.

As the user environment documentation matures, it will be reshaped into a tutorial. The tutorial will be made available to the public via LLNL’s hpc.llnl.gov website.

6 Deploying the LC Magic User Environment

With the El Capitan ATS deployment, LLNL has had the opportunity to revisit the way in which its user environments are deployed, not only for ATS, but for all of its other platforms as well. The new user environment deployment methodology is based on Spack, the open source HPC package manager. Over the past decade Spack has grown in both feature-set and popularity, not just for library and application developers, but for HPC facilities staff as well. For the past 2 years, LLNL has been developing the ability to build, test, and deploy its user environment with Spack.

6.1 Spack-Based Deployment

The LLNL user environment is deployed using Spack, specifically the Spack Stacks capability that enables building several software stacks from a single, cohesive environment. Spack allows us to provide a single environment definition that enables easy modification and extension of the environment as more compilers and packages are added. Spack also provides several mechanisms for the key portability aspect of the environment that gives us the ability to build the environment on a machine other than the target machine while also providing a means for defining the layout and organization of the packages for easier consumption by our users.

Spack capabilities have been augmented by the use of a wrapper shell script to drive the installation process and ensure a consistent installation with site- and machine-specific changes as needed. This script has been used to enforce consistent usage of Spack as well as checking for settings in the builder's environment that could have unintended impacts to the final installations. This script has developed considerably since this project began to include a host of fixes for issues discovered after installation as well as contributing to the portability of the environment by modifying aspects of the environment to appropriately reflect the network and machine on which the environment is installed.

The current environment build and deployment strategy involves building the environment on a test machine and then copying it to the final machines. This provides a mechanism for testing the build process and checking the final result before impacting end users. This also allows for a single environment to be built and deployed to multiple, similar systems, ensuring a consistent environment for users. An added benefit of the process is that we are able to update the environment in such a way that it has an effectively unnoticeable impact on running user jobs.

6.2 Testing with Goulash

Our critical Tri-Lab ASC codes and their build systems are often unique in their requirements for the development environment. Existing ASC open-source benchmarks have simplified build systems (so that vendors are willing to build them) and have been found to not be sufficient to test that a user environment has all the required functionality. The LLNL open-source project Goulash[1] was explicitly created to fill this gap. Goulash has pulled material from dozens of ATS and CTS user environment problem reproducers from the last 20+ years covering compiler, GPU, and MPI issues. The Goulash test suite represents capabilities critical for building ASC

[1] https://github.com/LLNL/goulash

codes. This suite has a track record of reproducing errors on multiple generations of ATS and CTS systems.

The Goulash test suite is built around a single-file application template written both in C++ and Fortran90. The application is templated such that variations of it can be generated to test desired CPU and GPU computation models: HIP, CUDA, OpenMP GPU offloading, lambda functions (needed for Raja and Kokkos), critical MPU communication patterns, and more. To exercise the various compilers (i.e., CCE, AMD ROCM, GNU) an interoperability template is used. Thusly, each combination of language, computation model, and compiler results in a single program which can be run to verify functionality, accuracy, and performance.

The Goulash test suite allows quick, straightforward verification that all desired functionality is working for each and every software release. Goulash is the first thing run before any compiler or MPI delivered from HPE or AMD is installed. The entire suite can be run in approximately 1 minute on 2 compute nodes. It often detects user environment issues that need to be addressed (either through workarounds or vendor involvement) before deploying the new software to users. When all Goulash tests pass, we have high confidence that the examined compiler/MPI release will work for our users. While the test suite is used by HPE and AMD to test their compilers, it is often the interaction of HPE, AMD, and GNU that can cause issues and Goulash continues to regularly find issues for EAS-3 software releases. With Goulash, dozens of problems have been reproduced and reported to our vendors partners.

6.3 User Environment Modules

Similar to the Cray environments (and indeed most HPC user environments), the LC Magic user environment relies on modules which users load to activate certain tools. The user environment modules exist in a hierarchy and only compatible tools can be loaded concurrently, shielding users from version or compiler mis-match issues.

To differentiate the LC Magic tools from the vendor-provided tools the word “magic” is appended to a module’s version number (e.g., cce/16.0.0-magic). In addition, the path to the compiler executable itself encodes the version, including the “magic” designation (e.g., /usr/tce/packages/cce/cce-16.0.0-magic/bin/crayCC). The paths to MPI wrappers indicated compiler, compiler version, MPI implementation, and version (e.g., /usr/tce/packages/cray-mpich/cray-mpich-8.1.26-cce-16.0.0-magic/bin/mpicxx).

Through the module system and the use of “-magic” for compiler versions, LC is able to leverage the module hierarchy. If a user switches from a vendor-provided compiler to the LC Magic version, all currently loaded tools are evaluated and compatible versions are reloaded into the environment.

6.4 Anticipating the Arrival of El Capitan

Looking forward to the arrival of El Capitan and its unclassified sister machines, the LLNL user environment deployment process is ready with many improvements still to come. We do not anticipate any major changes when deploying to El Capitan. We are also well situated to quickly deploy updated environments as more packages are requested. As such, even if changes are required upon delivery, our deployment strategy will enable a rapid response.

7 Livermore Development and Deployment Efforts for Particular Tools

This section details efforts made for discrete aspects of the user environment.

7.1 Compilers

Our huge ASC application often mix many physics and utility packages built with the AMD HIP C++ compiler, HPE CCE Fortran OpenMP GPU offloading compiler, plus many third party libraries compiled with g++ 10.3.1 in order to create one big physics application. The build systems used are very complex and mismatched compiler versions can create nearly impossible to track down application problems. For this reason LLNL pushed HPE to create module-less compiler and MPI environments described above to support Spack and LLNL's development environment. LLNL also provided the "magic" environment (also described above) that makes building huge ASC application with exactly the desired compilers and MPIs straightforward. With the Goulash test suite (Section 6.2), new compiler and MPI installs can be fully vetted in just a few minutes.

That said, with every compiler and MPI release, existing issues are verified as fixed and new issues are found by the users. Users report issues using the COE Jira systems (Section 5.1.3) and the COE staff, working with LLNL's development environment group, works with the code teams to identify bugs, find workarounds, reproduce the issues, and work with vendors to get fixes put into future releases.

7.2 MPI

With regards to MPI, it was uncovered early on that changes were necessary to get HPE Cray MPI to launch correctly with Flux. Link conflicts between the included Process Management Interface (PMI) library included with HPE Cray MPI, the PMI library included with Slurm, and the Flux PMI library implementation had to be factored into development environment builds, but the selection of library was only the beginning. The Flux team at LLNL engaged in a considerable effort[2] to ensure HPE Cray MPI launched correctly under the Flux scheduler.

Collaborations with Open MPI and the support vendor for MVAPICH (X-ScaleSolutions) to get compatibility with the Cray Slingshot 11 Interconnect have continued under the auspices of the COE, with Open MPI supporting both the interconnect and accelerated APU transports (but with performance debugging still necessary) and MVAPICH supporting the interconnect with their MVAPICH 3.0a release, but not yet supporting the accelerated APU transports.

Finally, and most recently, the TOSS team has made a working XPMEM library available, which has shown a significant increase in on-node MPI performance in preliminary benchmark testing.

7.3 Debugging Tools

Once codes are built on a new platform, the ability to debug (i.e. diagnose and remedy code errors) becomes a critical capability. Early collaboration with AMD and HPE led to the enablement of ‘printf’ debugging on the AMD GPU, where the GPU kernel does NOT need to be completed for the messages to be received. This is critical for diagnosing ‘hang’ states and is

[2] https://github.com/flux-framework/flux-coral2

something that was desired, but missing, on CORAL1). LLNL's long-standing preferred interactive debugger tool TotalView has also been adapted to AMD GPU debugging on EAS3 in collaboration with HPE and AMD. A further example of collaboration with vendors can be seen with the STAT tool. ~~Something something~~

7.3.1 STAT

The Stack Trace Analysis Tool (STAT) was developed primarily at LLNL for debugging application hangs. ~~Over the years it~~It provided~~has proved~~ invaluable in debugging hangs even at ~~the most~~ extreme scales of over a million processes on the Sequoia Blue Gene Q system, which would be far beyond the capabilities of other debugging tools. STAT has been ported to run on El Capitan systems. This includes the ability to use the rocgdb debugger as a backend to gather stack traces from AMD GPUs, and also required a~~a~~ port to be able to debug applications launched with the Flux resource manager. LLNL has also worked with HPE (previously Cray) to integrate STAT into their native programming environment. This ensures that a validated STAT installation will ship with El Capitan systems without additional LLNL deployment efforts. A~~Furthermore, a~~n official HPE programming environment installation provides for~~a~~ deployment at other sites, furthering ~~increasing~~ the tools adoption and robustness.~~.~~

7.4 Performance Tools

Once codes are building reliably and generating the correct answer, code team efforts turn to performance tuning. This often continues intensely for several years after the arrival of ATS machines. Both HPE and AMD provide a proprietary suite of tools for performance tuning. The COE helped many code teams use these performance tools, especially through hackathons, to significantly improve the performance of key sections of their codes. This experience has led to many improvement suggestions for the tools. Several of these hackathon-based suggestions have already been delivered and many more are scheduled to be delivered in the coming year.

In addition to the vendor provided tools there is a strong desire for third-party and open source performance tools to provide portability across architectures and to leverage expertise developed in academia and the HPC user community. An example of additional non-vendor performance tools is described below with TAU and HPCToolkit.~~Something something~~

7.4.1 TAU

TAU is a performance analysis toolkit that provides profiling and tracing functionality on a wide range of computing platforms. LLNL has maintained a consistent support and training contract with TAU developers and has made use of this relationship to develop and provide TAU to application developers on new LLNL platforms. In regard to the CORAL-2 development environment, TAU supports performance data collection for AMD profiling and tracing libraries rocprofiler and roctracer. TAU developers have been updating support for the most recent ROCm releases to address changes in the rocprofiler API. A TAU representative will be holding an on-site workshop at LLNL on August 24th, 2023, to include training for using TAU on LLNL EAS-3 systems. This workshop will present examples using an updated TAU with a recent ROCm release.

7.4.2 HPCToolkit

The HPCToolkit team from Rice University has been plugged into the CORAL-2 Tools Working Group through a TriLab CCE contract. They have worked with both HPE and AMD to driver requirements, design tool interfaces, and test performance tool capabilities. With a significant amount of feedback from Rice, AMD will soon be releasing a new version of the ROCm perf tools interface that will allow attributions of performance to individual source-code lines that are part of a kernel.

Working with LLNL, HPCToolkit is also being prepared to work on the large, complicated codes that will be run on El Capitan. The tool is tested at deployment time against applications from each laboratory, and numerous improvements to HPCToolkit have been driven by the unique challenges found in TriLab codes.

7.5 Memory Correctness Tools

One of the most critical correctness tools for ASC code developers are memory corruption detectors, both on the CPU and on the GPU. On the CPU, address sanitizer for clang-based compilers, known as ‘asan’, is widely available and used. For the AMD GPU, ROCM's memory corruption detector is currently under development and is also based on clang's address sanitizer. As part of our collaborative effort with AMD, AMD provided ~~us~~ an alpha release of the GPU address sanitizer in March 2023. We installed it and ran our benchmarks and several user apps through the alpha release. This alpha release found several undetected and real off-by-one array addressing bugs in our GPU benchmarks. This alpha testing also revealed several shortcomings of the current release which LLNL reported. ~~and~~ AMD has indicated these will be fixed in the upcoming beta release. We are still many months out from the targeted official asan GPU support.

8 Ongoing Efforts

Although most, if not all, ASC applications have a good path for running on El Capitan at delivery, there are still more than 100 open tickets for issues that are requiring code developers to use workarounds to currently run. LLNL will continue to push for official solutions to these open tickets, make fixes available to our users in a timely fashion, file new issues to occur in new releases, and continue to evolve our development environment to facilitate the building, debugging, and tuning of complex ASC applications.

Additionally, the following user environment features need ongoing attention.

OpenMP 5.2 support is still being implemented in the CCE and AMD compilers. Through the Compiler Working Group, LLNL (and ORNL) continue to provide prioritization for implementing OpenMP 5.2 features. In our last review (as part of a NRE milestone), LLNL believes CCE and AMD have implemented all the high-priority OpenMP features our ASC code developers need. HPE and AMD continue to work on the low-priority OpenMP features that are currently missing.

LLNL continues to work with HPE on better integration of Cray MPICH with Flux, better support of alternative non-HPE MPIs on the Cray Slingshot fabric (to provide redundancy), and

to better support high-performance execute-only applications (no read/write bits set) via XPMEM that many ASC teams provide to their users.

AMD GPU profiling, tracing, and debugging tools are designed for serial programs, not MPI-based applications like ASC develops. To facilitate use of these serial tools on parallel programs, LLNL provides our users with a script harness that allows users to run any command on exactly one task of the user's choice. Usually rank 0 or a known ‘problem’ rank is used with this script.

9 Conclusion

LLNL’s user environment for El Capitan is in very good shape and we are well prepared for the delivery of the first El Capitan nodes. From past experience with many ATS deployments at LLNL, we know that the first two years after delivery of a new ATS platform involves intense debugging and tuning efforts by the ASC code teams. We are well prepared with solid and well-utilized issue tracking capabilities, robust methods for fast rollouts of new software releases into the user environment, hackathons to support intense development and debugging efforts by ASC code teams, and several well-tested methods for accessing the development environment in a way that best suits each code team’s needs.

Appendix A

Letters of Support

LLNL ALE3D Code Team

See the following pages.

LANL Eularian Applications Project Team

See the following pages.

SNL EMPIRE Code Team

See the following pages.

LANL Ristra Code Team

See the following pages.

SNL SPARC Code Team

See the following pages.

Appendix B

User Documentation Website

The following pages are an export of the “El Cap EA Systems” documentation wiki site from August , 2023.

Appendix C

User and Vendor Presentations

March 2023 User Environment Deep Dive

The following pages contain the slide deck for the presentation to vendors. In particular, this presentation is a deep dive into the philosophy behind the LC Magic user environment.

What is Flux and why do I care?

The following pages contain the presentation given to COE users on March 28, 2023.