Building the Tri-Lab Computing Environment

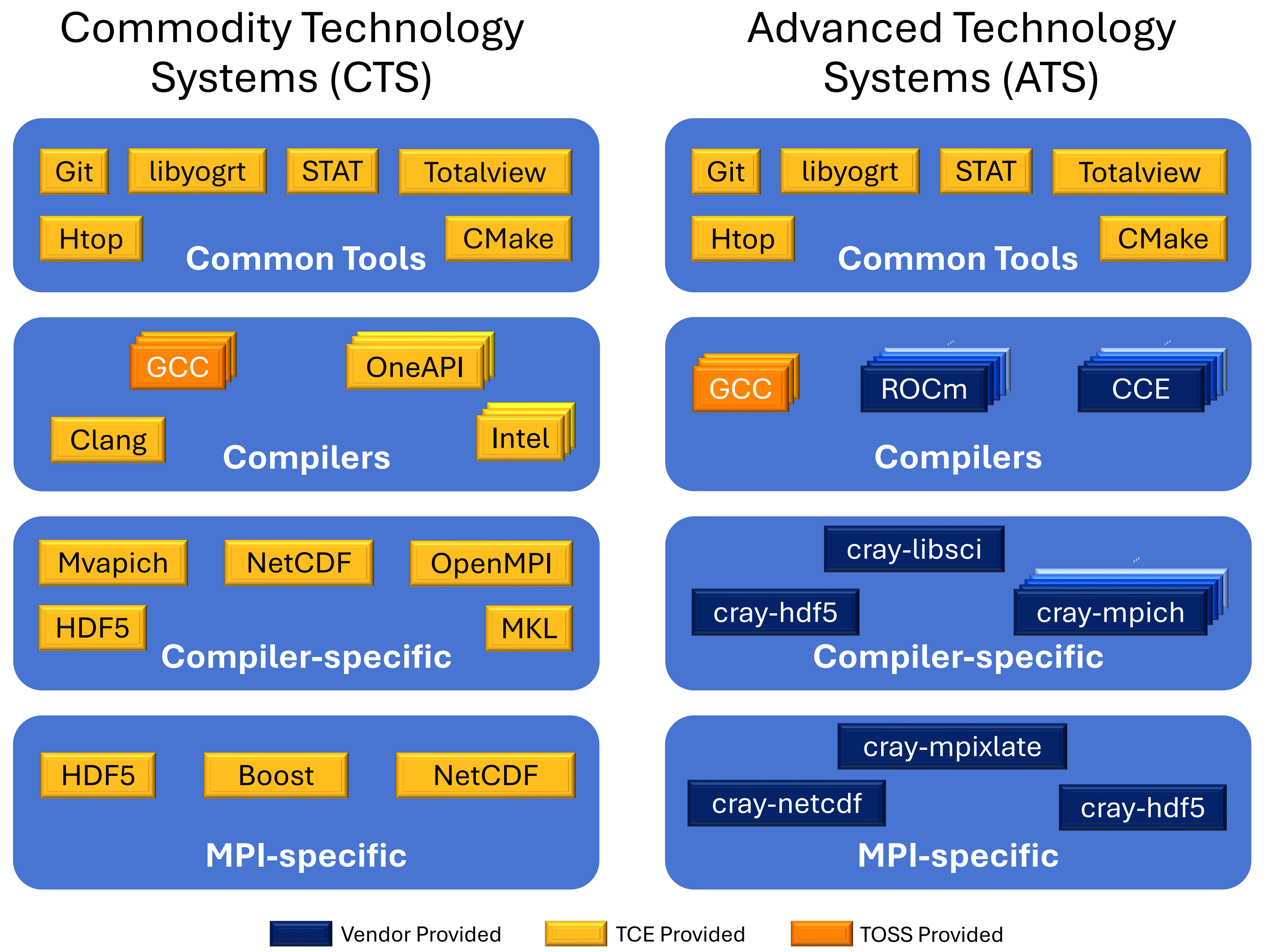
A fast, robust, and collaborative build process

Nicholas Sly (LLNL)

The **Tri-Lab Computing Environment (TCE2)** is a matrix of compilers, MPI implementations, scientific libraries, visualization applications, debuggers, profiling tools, and other commonly used applications, totaling over 800 packages and dependencies. These are built on top of the software provided with TOSS and the vendor-provided programming environment. TCE2 strives to provide a common programming environment across the three NNSA labs.

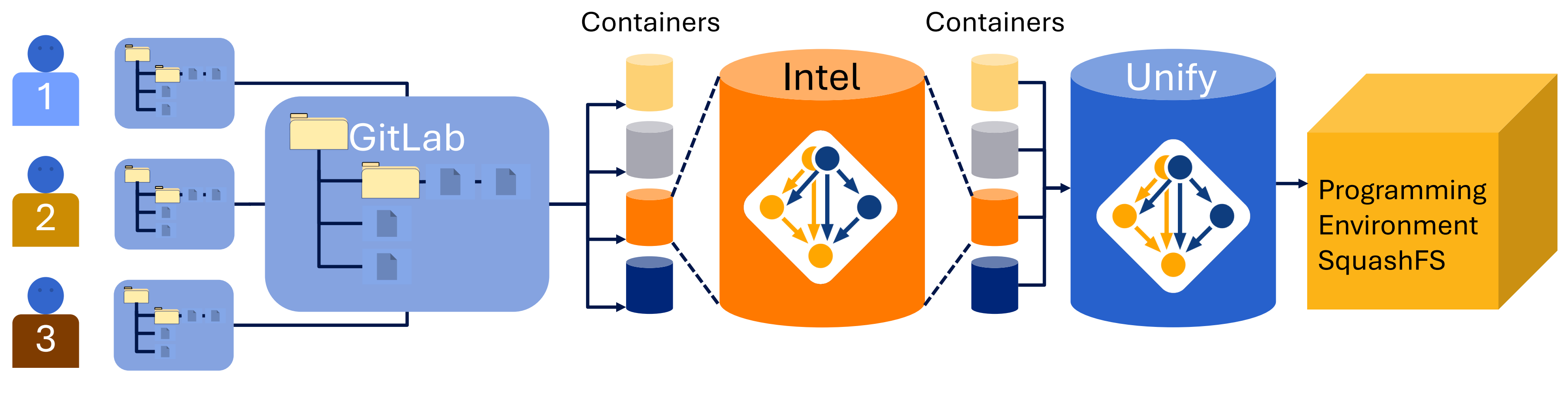
What Does TCE2 Provide?

Because ATS machines (HPE/Cray) come with a rich programming environment, fewer packages need to be built with TCE. Our CTS machines require more packages be provided.



TCE provides the vast majority of software for the CTS programming environment but only a subset on ATS machines, due mostly to vendor-provided software.

Improved Process Enables Better Delivery



Programming environment contributors push their environment changes to GitLab, which kicks off a series of containerized CI jobs to build individual elements of the environment. The environment is then collected into a single build and compressed into a SquashFS file, which can then be distributed to the relevant machines and installed.

- Source version-controlled via GitLab for **easy multi-user collaboration**
- Continuous Integration (CI) **builds are performed automatically**
- Containerized and distributed, it allows **faster, system-independent builds**
- Installation changes made **specific to the package** affected by the change

Old Processes Were Disjointed and Fragile

Before TCE:

- Team of individuals responsible for package installations
- One person per package
- Complex dependencies could require a serial person-by-person process
- Scripts for installs
- Manual module writing
- Installations placed in a moderately organized, shared network space
- Modules mostly handwritten and copied with slight revisions for new versions

Since TCE (at LLNL):

- Spack builds the packages and modules
- LLNL-specific changes made after Spack builds packages
- Environments compressed and distributed to each machine
- Machine-specific changes made on target machine

Pain points of existing method:

- MUST run on a specific machine
- MUST change to a specific user
- MUST run a specific set of scripts
- Modifications to build process extend script (4000+ lines)

Users Benefit from Faster, More Portable Builds

- Spack buildcache
  - Reduce duplicate builds
  - **Greater commonality** across code dependencies
  - **Faster build times** for users
- Containerized Spack stacks
  - Build **system-compatible applications**
  - Provide **portable builds** with a **familiar programming environment**
  - Integrate into user build processes to **streamline development and deployment**
- Programming environment testing
  - Elements can be **tested as part of the CI pipeline** for producing the environment
  - Catch regressions in vendor software or our software
  - Verify resolution of user issues

Conclusions

The newly designed TCE build system allows for improved programming environment creation and upkeep

- Much less facility-specific knowledge required for contributing
- Leveraging Tri-Lab and wider community open-source software

Next Steps

- Further reduce post-install script by integrating changes into Spack
- New methods of providing tools to users:
  - Containerized environments
  - Spack binary cache
- User-defined programming environments
- Portable environments for consistent user experiences across machines, networks, and HPC centers

A modernized TCE build strategy reduces maintainer time investment and enables bonus user benefits

Lawrence Livermore National Laboratory

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-POST-2004738

NNSA National Nuclear Security Administration