Expediting Research with SPACK

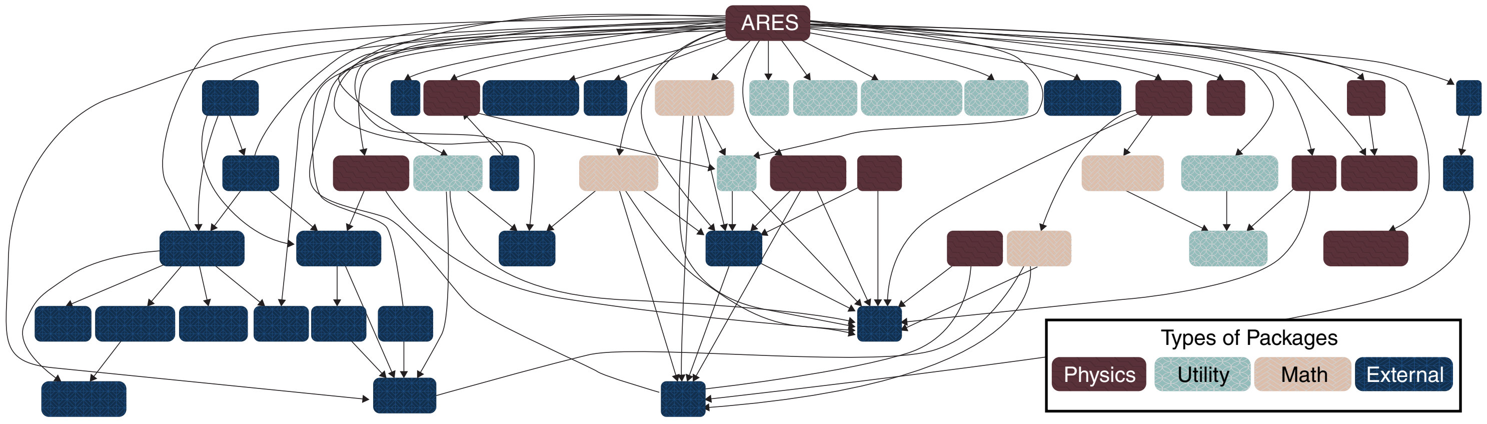
IN early 2013, Todd Gamblin was a staff scientist at Livermore mentoring graduate students as they struggled to build and integrate software on high-performance computing (HPC) systems. Seeing days and weeks of valuable research time lost to getting computing programs off the ground, Gamblin thought to automate the task of building and installing the disparate software elements HPC users required.

The solution, Spack (for Supercomputing PACKage manager), would go on to expedite operations across a community including top supercomputing centers, cloud providers, and individual users worldwide. "Fundamentally, Spack ensures all the software used by an HPC system is built in a compatible and efficient way," says Gamblin, now a distinguished member of technical staff in the Livermore Computing Division.

Laying Computing Foundations

Building software—that is, turning software code into usable applications—is a notoriously difficult computing challenge in its own right. Scientific programs used by HPC centers rely on a constellation of interdependent software packages that only function if software versions, languages, feature options, and other configuration options are properly coordinated.

"HPC users had done these tasks manually for many years," says Gamblin. "While package managers existed for other domains, the fine-tuning needed for HPC systems led most users to build software by hand. The software build process is painstaking and error prone even for experts." Since laying its foundation a decade ago, Gamblin's automated approach to building software has graduated from personal to global use. Recognizing potential to support stockpile stewardship, the Laboratory designated funding from its Advanced Simulation and Computing (ASC) Program to continue Spack's development. In 2019, the project received an R&D 100 Award (see *S&TR,* July



Spack automates the process to download, install, and manage scientific software packages. The application resolves package dependencies and user preferences to ensure operability of software stacks in ways not possible with manual integration. For example, Spack automates the build and deployment of the 46 dependency libraries (represented by interconnected boxes) that comprise ARES, a hydrodynamic simulation code used for fusion experiments at the National Ignition Facility.

2020, pp. 10–11), fueling even wider recognition and adoption within the HPC field.

Spack's automated approach to building, testing, and deploying software has reduced project software distribution times from several weeks to hours or even minutes, depending on the size of the stack. As preeminent computing systems (such as early access systems for Livermore's upcoming El Capitan exascale machine) come online, Spack ensures that users continents apart who employ different software, write in different programming languages, and execute asynchronous tasks can leverage each other's collective work to use HPC resources.

"Most people want to use a particular tool but don't care about its specific version or build options. However, getting the version right is crucial; a single incompatibility between packages could cause a project to fail to build," says Gamblin. With Spack, users can now create multiple versions of code and ensure compatibility before final deployment, averting experimental delays by preemptively resolving any inconsistencies. For example, Spack supports Livermore's proprietary radiation hydrodynamics codes, ARES and HYDRA, which aid inertial confinement fusion experiments at the National Ignition Facility (NIF) by incorporating dozens of software packages for modeling lasers, electromagnetic fields, and plasma physics. Purpose-built to model these experiments, HYDRA was used to design the December 5, 2022, NIF shot that achieved fusion ignition.

An Open Approach

An open-source project from inception, Spack is freely available on GitHub and actively seeks contributions. "The Spack community works to ensure that packages are integrated and kept up to date. When other organizations want to contribute updates to recipes, we welcome that. We benefit from the updates, and they benefit from having the needed changes integrated into Spack in a sustainable way so that future updates continue to include their revisions," says Gamblin. Other HPC sites benefit as well. For instance, deployment time for an extensive 1,300-package software stack on Oak Ridge National Laboratory's (ORNL's) Summit supercomputer was slashed from two weeks to 12 hours. The same stack has been ported and built on ORNL's Frontier system, the world's first exascale machine.

By enabling and expediting scientific research, Spack has cemented itself as an invaluable tool for the Exascale Computing Project (ECP) within the Department of Energy (DOE). ECP provides the next-generation computing software stack necessary to tackle challenges of national importance—including clean energy generation, stockpile stewardship, and materials discovery—to which Lawrence Livermore contributes significant computing resources.

What began as a library of roughly 100 software packages has grown into a trove of more than 7,000, contributed by researchers from other DOE laboratories, academic research institutions, industry, and international collaborators. The roughly 100,000 lines of code that comprise Spack's "core" are maintained by Lawrence Livermore and trusted members of the Spack community. Most of the remaining 250,000 lines originate from outside Livermore and form an expanding package repository for assembling software. Today, the community boasts more than 1,200 contributors and many thousands of users.

Rapid growth demands resources and external collaboration. While a handful of dedicated developers at the Laboratory manage core feature development and community contributions, extended core teams at subcontractors Kitware and TechX conduct build farm maintenance, Windows support, and continuous integration automation. Among GitHub contributors, roughly 40 users are trusted package maintainers authorized to merge pull requests; a broader set of more than 300 users helps to review submitted package recipes.

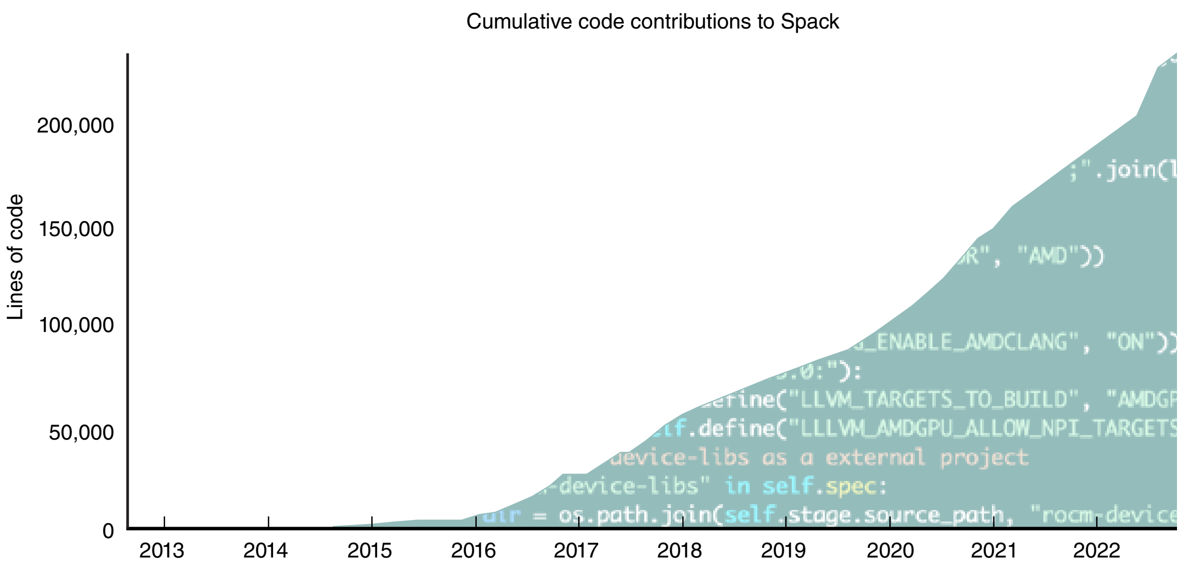
Software developer Greg Becker maintains Spack's functionality and supports its user base. In response to the tool's surge in growth, Becker and his colleagues spend significant time assessing code contributions and managing bug fixes to maintain consistency within Spack's repository. "Keeping up with all that growth is a challenge," Becker says, "but the great thing about open source is that anyone can jump in and help on a part of the project they care about." Despite the demands, Becker says, "Without all the work from the community, Spack wouldn't be nearly as capable as it is. The community is a huge force multiplier."

Spack's collaborative roots have spurred numerous partnerships. Amazon Web Services (AWS) invests significantly by providing cloud computing credits for Spack's community, enabling automated build testing and publication of a large cache of pre-built software. AWS has also hosted Spack-centered tutorials and hackathons with Livermore team members. Semiconductor developer AMD has contributed to ROCm™ software packages and the compiler that translates source code into machine code from high-level programming languages into computer-legible form. Intel helps to maintain and support packages for its oneAPI programming model. NVIDIA supports its NVHPC compiler recipes. HPE Cray performs continuous integration for Spack packages in its own supercomputing environment. Other open-source frameworks for software benchmarking and testing have emerged. For example, Ramble, written by Google Cloud staff, uses Spack to build benchmark programs, but adds a harness for running experimental performance campaigns. And in Japan, Spack was chosen as the deployment tool for the world's second-fastest supercomputer, Fugaku, manufactured by Fujitsu and Riken.

Current and Future Challenges

At the 2022 International Supercomputing Conference, the Spack team announced the latest release, v0.19.0, with a redesigned "concretizer," the component that resolves dependencies, versions, and other preferences for a software build. "Getting all the software packages to agree with one another is an NP-hard combinatorial problem," says Gamblin. NP-hard problems are the equivalent of searching for a needle in a haystack; an optimal solution must be located among millions of possible software configurations. The best known algorithms amount to guessing-and-checking solutions in the worst case, but they leverage sophisticated search techniques such as Conflict-Driven Clause Learning and multi-objective optimization to zero in quickly on good solutions. Nearly all package managers must contend with dependency resolution. "Like a game of Sudoku," says Gamblin, "Spack uses sophisticated criteria to make an educated guess, identify possible compatibility conflicts, and backtrack to quickly find a valid solution."

Resolving installation parameters is only the beginning. Computing's Tammy Dahlgren, a core Spack team member, puts the issue simply: "Just because you can build it, that doesn't mean it will run correctly." Dahlgren supports the ECP-funded Extreme-Scale Scientific Software Stack (E4S), which uses Spack to facilitate deployment of around 600 HPC software packages directly from source. "Once a package has been installed, E4S ensures the software remains functional while



```
                                       ;".join(l
                             R", "AMD"))
                  _ENABLE_AMDCLANG", "ON"))
           .0:"):
        efine("LLVM_TARGETS_TO_BUILD", "AMDGP
     lf.define("LLLVM_AMDGPU_ALLOW_NPI_TARGETS
   device-libs as a external project
  -device-libs" in self.spec:
 dir = os.path.join(self.stage.source_path, "rocm-device
```

Spack's open-source nature enables contributors worldwide—including Department of Energy national laboratories, industry, and academic institutions—to utilize Spack's resources and offer functional additions, accelerating support of new use cases. Contributions to Spack's GitHub repository over time (excluding those to the core) have dramatically increased in recent years.

Development team for Spack: (from left) Greg Lee, Matt Legendre, Todd Gamblin, Peter Scheibel, Greg Becker, and Tammy Dahlgren. (Photo by James Chalabi.)

underlying systems and package dependencies are constantly updated," she says.

Spack not only optimizes software build configuration, but also tests to ensure builds will operate as intended within a larger system before executing the immense task of deploying HPC programs. Utilizing "test recipes" included with software packages, Spack can assess package functionality in different configurations. Test recipes themselves can be customized to different versions and configurations. "That way, tests evolve with the software and adapt to the installation," says Dahlgren.

Although Spack has already expedited projects on several supercomputing systems, accommodating the complexity of Lawrence Livermore's El Capitan system—slated to be the world's most powerful supercomputer—is no easy feat. "The software stack for advanced supercomputers keeps getting more complicated, and some of the complexities of El Capitan will push the boundaries of what we can model with Spack," says Becker. The combination of compilers and other low-level tools necessary to meet heightened demands will require Spack to maintain consistency across distributed computing resources. To address the challenge, Becker says Spack's software package models continue to mature.

As an adaptable open-source tool, Spack is also well suited to bridging the gap between HPC and cloud-based workflows, the latter used to expand, diversify, and distribute computing loads. However, Becker explains, "HPC workflows are designed with static resources in mind: something is built once and meant to run for years. Cloud architectures instead focus on more ephemeral resources." Shorter lived systems can lean on Spack's ability to install quickly from binary caches rather than building from source. "Ultimately, we want an arrangement that rapidly gets us from a blank slate to doing science," says Becker. The team is currently working with scientists who use cloud resources for HPC to help them take advantage of the flexibility of cloud computing without forfeiting valuable research time to previously solved software installation problems.

"Spack's open approach to development is critical to its longevity," says Gamblin. "Large community efforts generate momentum, making the project easier to sustain in the long run." Dahlgren echoes Gamblin's view: "Spack is my first open-source software project, and I'm glad I joined. Interacting with talented people both at the Laboratory and across the world has been incredibly fulfilling."

–Elliot Jaffe

Key Words: Advanced Simulation and Computing (ASC) Program, concretizer, El Capitan, Exascale Computing Project (ECP), Extreme-Scale Scientific Software Stack (E4S), GitHub, high-performance computing (HPC), open-source, software package manager, Spack, supercomputing.

For further information contact Todd Gamblin (925) 422-9319 (gamblin2@llnl.gov).