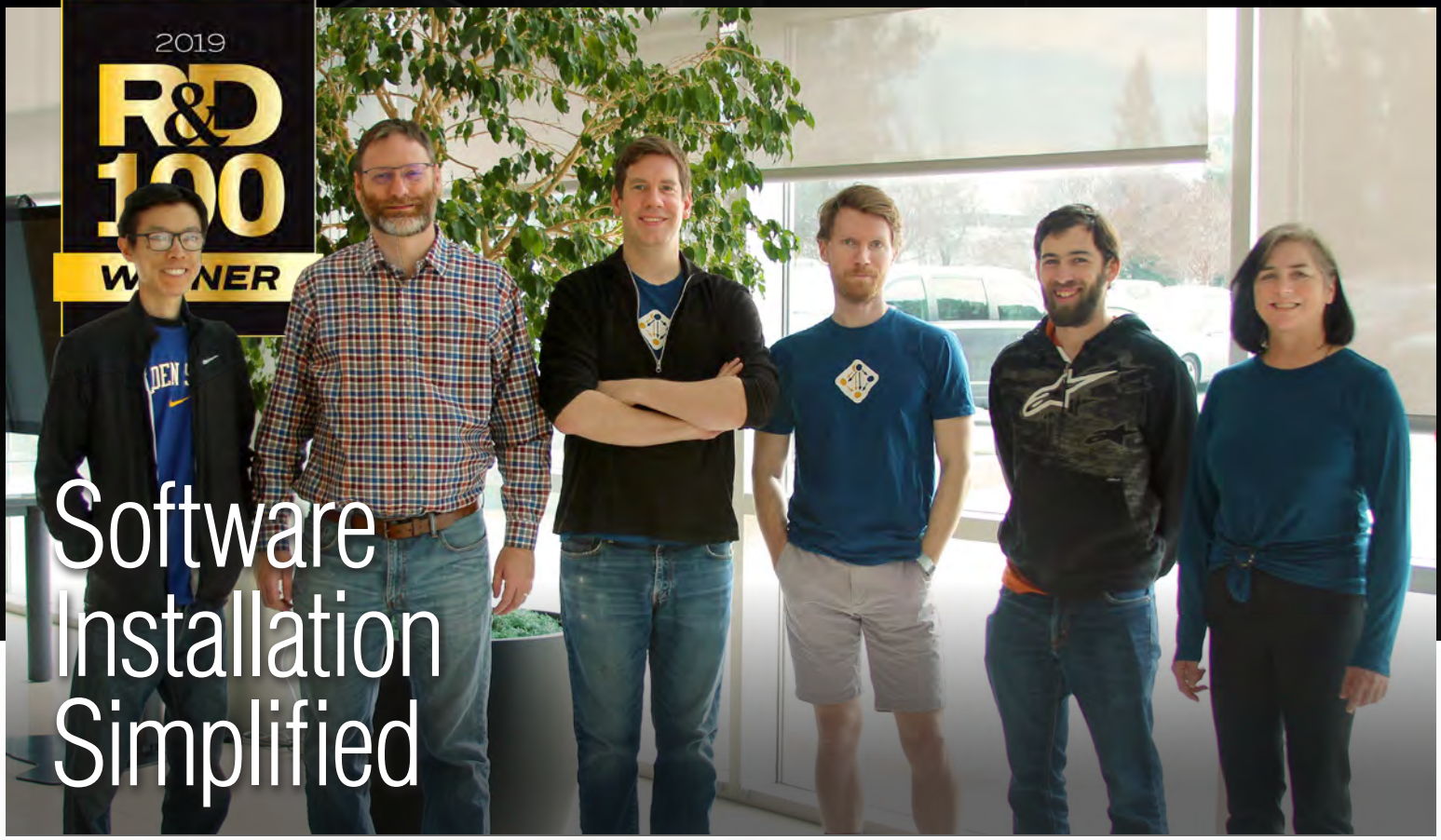


Software Installation Simplified

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

HIGH-PERFORMANCE computing (HPC) centers, such as Lawrence Livermore's, routinely deploy large-scale scientific software applications. These applications rely on tens to hundreds of external programs, known as packages, that enable the computer's operating system to execute specific functions and perform as needed. Packages include software versions tailored specifically for users' machines as well as dependencies and configurations customized for their applications. Even for seasoned professionals, manually downloading, installing, building, and resolving conflicts among all of these programs is onerous, and this time-consuming process is a substantial barrier to scientists.

Spack, a 2019 R&D 100 Award–winning software technology, was created to alleviate the package managment burden. "The inefficiency and complexity of building software for HPC machines can detract from scientific research," explains Livermore computer scientist and Spack principal investigator Todd Gamblin. "Spack's automated package management eliminates much of the grunt work." Codeveloped by a dozen other universities, national laboratories, institutes, and computing centers, Spack also won a silver medal in the R&D 100 Awards' Market Disruptor category.

Gamblin notes that other package managers can only be run by administrators and other privileged users and do not handle simultaneous custom installations of multiple software versions and configurations. Fortunately, Spack is designed for these scenarios and can be used by nonprogrammers, developers, and system administrators alike. Users need only download the Spack tool and learn its specification syntax.

Complexity Behind the Scenes

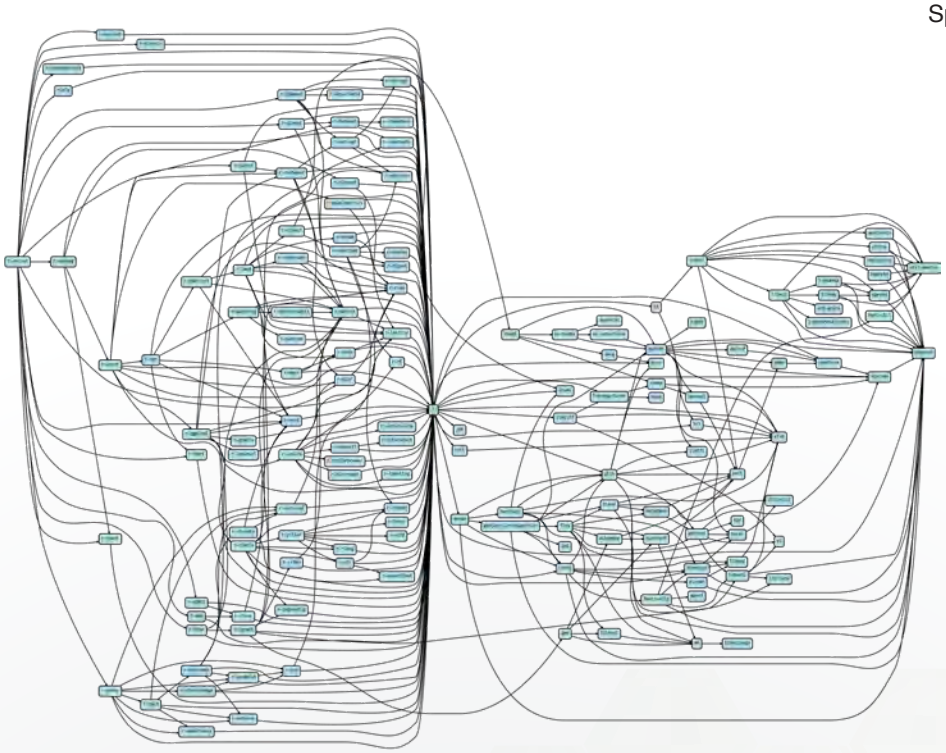
Installing and using software dependencies is a careful balancing act as new packages are integrated into an application and each addition increases the complexity of the task. Spack speeds up installation by assembling all the packages needed for an application's deployment, managing their configurations, and optimizing the build for the user's machine. The user views and queries Spack's list of available packages, then Spack automatically downloads and builds source code for the desired packages' dependencies. Users can assemble hundreds of software libraries in minutes, giving them more time to focus on their scientific research.

Spack's recipes—steps for building a package—are written in the widely used Python programming language. On top of Python, Spack provides its own domain-specific language that enables contributors to write templated instructions, so a single recipe file can be used to build many different configurations. (Other package managers can require thousands of duplicate files to accomplish the same task.) To build a configuration, Spack offers a custom specification language for selecting

options, versions, and compilers. Gamblin states, "Together, Spack's package recipes and specification syntax allow users to tailor their software stack for specific codes and computing environments."

Spack's concretization algorithm is responsible for converting the user's abstract requirements into concrete, buildable specifications. The process effectively fills in the blanks of software configuration. The user provides a partially completed form, and the algorithm finds a configuration that satisfies the user's requirements as well as each package's unique compatibility rules. The algorithm produces an output file of the resulting configuration data, allowing users to easily reproduce a software stack for a particular scenario.

Perhaps Spack's most impressive feature is its repository of thousands of templated packages supporting diverse computing platforms (including laptops), simulation frameworks, programming languages, and other options. Gamblin explains, "A user's software integration burden increases with every new library or update, but Spack manages the growing complexity and allows users to build quickly on a variety of computing systems."



A typical multiphysics code used at Lawrence Livermore requires installation of hundreds of software packages and dependencies, as illustrated by this map of the "rminer" package. Whereas manual installation of such complex software would be impossible, Spack's automation makes quick work of the task.

Worldwide Impact

Spack's features and flexibility have led to its adoption by many prominent coding teams, supercomputing centers, and software development communities at Livermore and beyond. For example, Oak Ridge National Laboratory uses Spack to deploy more than 1,300 software packages on the top-ranked Summit supercomputer. This installation process previously required two weeks of work and can now be deployed overnight. Spack is also used at Los Alamos National Laboratory, Fermi National Accelerator Laboratory, CERN (the European Organization for Nuclear Research), and the Japanese research center RIKEN.

The Spack team regularly offers tutorials and workshops at major supercomputing conferences and visits HPC centers to learn from and train development teams. Spack is used for software deployment on 6 of the world's top 10 supercomputers and has been adopted as the standard deployment tool of the Exascale Computing Project (ECP)—a Department of Energy collaboration tasked with building a reliable software stack for future exascale-class machines. According to ECP deputy director Lori Diachin, "Spack is an integral part of the ECP's ecosystem because our software stack is quite large and complex."

Spack is open-source software, which means its functionality can expand and its features can mature thanks, in part, to the software community beyond the Laboratory. (See also the article on p. 8 of this issue; *S&TR*, January/February 2018, pp. 4–11.) Spack has more than 500 contributors and 2,000 monthly active users around the globe—and the numbers are growing."Learning about use cases at research institutions and other HPC centers has helped us make Spack what it is today," says Gamblin. "Open-source development benefits those who use and contribute to Spack."

Next-generation HPC architectures, with diverse graphics processing units and accelerators, will only increase the complexity of scientific applications and the necessary software dependencies. As the bar is raised, Gamblin expects Spack to become smarter and more automated. He states, "We are always expanding Spack's capabilities to adapt to new technologies and user needs."

—Holly Auten

Key Words: compiler, concretization algorithm, Exascale Computing Project (ECP), high-performance computing (HPC), open-source software, package, package manager, R&D 100 Award, software installation, software stack, Spack.

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