Resiliency in Computer Applications

THE stakes are high when scientific applications run on high-performance computing (HPC) systems. Simulations of complex phenomena such as fusion energy and natural disasters require timely analysis to deliver effective solutions. "Checkpointing" techniques help protect against computing failures that slow performance by periodically saving application data as the simulation progresses. When a crash occurs, the application is restarted from the checkpointed file instead of from the beginning—analogous to the way a word-processing program autosaves and subsequently recovers a document.

In traditional checkpointing, application data is saved to and retrieved from the computer's long-term storage—actions known as input/output (I/O) operations. The application waits during this potentially minutes-long process, unable to continue its computations. Researchers must regularly monitor their applications and manually intervene if failures occur to avoid costly slowdowns. "The time to reach a solution matters because scientists need to make time-critical decisions," explains Lawrence Livermore computer scientist Kathryn Mohror, who co-leads the development team behind the R&D 100 Award–winning technology known as the Scalable Checkpoint/Restart (SCR) framework.

Conceived in 2007 by Laboratory computer scientist Adam Moody when a large-scale simulation code failed repeatedly on the Atlas supercomputer, the SCR framework is a multilevel checkpointing system that alleviates the bandwidth bottleneck by caching checkpointed files in storage located close to the compute nodes. The framework leverages short-term storage locations, accelerates I/O operations, and creates failure-resilient checkpoint and restart support. Mohror, who has co-led the project with Livermore computer scientist Elsa Gonsiorowski since Moody stepped down from the position in 2019, says, "Simulations can be completed more quickly with a smarter checkpointing system like SCR."

Full Feature Set

On a supercomputer, checkpointed files are saved in a complex, hierarchical storage architecture. SCR manages the movement of checkpointed files through the storage hierarchy to achieve the best performance for the application. This process involves executing many complex tasks while the application runs—performing health checks of the computing environment, monitoring the application's progress, managing I/O data, and transferring data—and exploits storage levels that are not shared across all of a supercomputer's nodes. In addition, SCR saves only the checkpointed data needed at the time, not the entire system state. "In most cases, SCR can restore a checkpoint from short-term storage because most failures affect only a small part of the application at a time," notes Mohror. SCR can also fall back to checkpoints in long-term storage, if necessary.

SCR is further differentiated from other checkpointing tools by its I/O management techniques, including the

Development team for Livermore's Scalable Checkpoint/Restart (SCR) framework: (from left) Bronis de Supinski, Kathryn Mohror, Tony Hutter, Elsa Gonsiorowski, Greg Kosinovsky, Cameron Stanavige, and Adam Moody. (Not shown: Greg Becker and Kathleen Shoga.) (Photo by Randy Wong.)

types of output files it handles. Mohror explains, "With our latest software release, SCR manages more than checkpointed files to storage. Now, it also manages general files containing other simulation data." SCR's I/O mechanisms scale linearly with the number of compute nodes used by the application and are as much as 1,000 times faster than I/O operations that do not use SCR. This enhancement allows researchers to output higher resolution data more frequently from their application runs, leading to a better understanding of the results.

Moreover, SCR accommodates different types of HPC storage architectures and their resource management configurations. Mohror says, "For each storage device, system administrators can specify the device's size, its failure characteristics, and how many checkpoints to store before deletion." SCR's checkpointing mechanisms "wrap" around the code, independent of the device or operating system. No two supercomputers are alike, and such software portability techniques are crucial for adapting Livermore's codes to future exascale-class machines.

From the user's perspective, SCR offers a flexible application programming interface that easily integrates into an application's existing I/O code. The user specifies a few parameters that tell the computer when, where, and how often to capture checkpoints. No other code modifications are needed. Ultimately, the user does not need to understand or manage the computer's specific storage hierarchy and can instead focus on the scientific application.

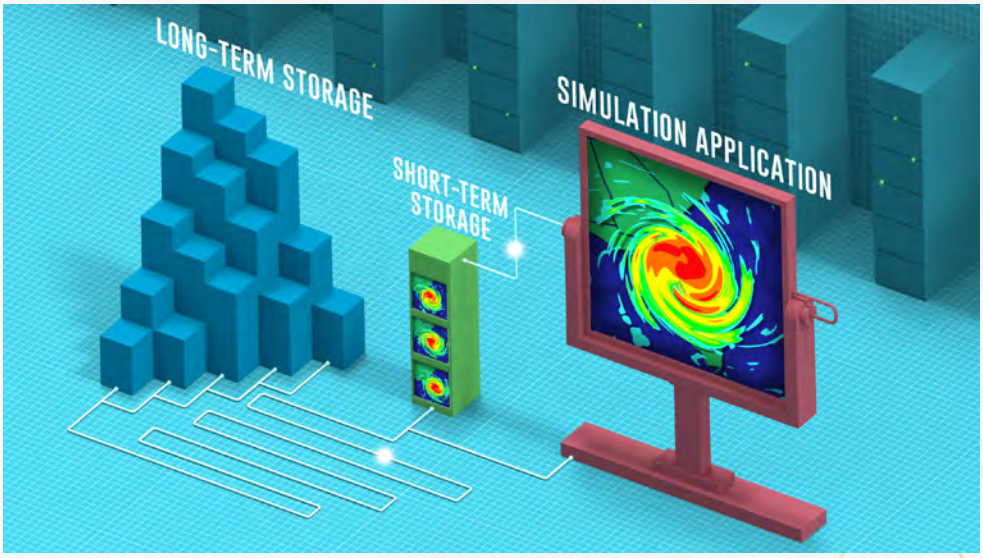
Proof in Production Codes

Livermore's pF3D code, used by the National Ignition Facility to simulate backscatter from laser light, was the first production code to use SCR. When a new supercomputer came online in 2007, Moody explains, "Each pF3D calculation needed days to complete, but the system failed every few hours. SCR saved and protected each checkpoint against system failures using data redundancy encodings." SCR reduced pF3D's checkpoint and restart time from more than 10 minutes to just seconds, allowing checkpointed files to be saved more frequently. Livermore physicist Denise Hinkel recalls, "I had been setting alarms to check the pF3D simulation periodically throughout the night, like it was a newborn baby. SCR's automated checkpoints and restarts let me sleep again."

SCR provides faster checkpointing, faster restarts, and portability across computing platforms. "We see orders of magnitude improvement in performance when using SCR, especially when running extremely large-scale applications," states Mohror. Such improvement is especially important as existing codes are modified for next-generation machines. For example, SCR sped up pF3D's checkpointing time by 48 times on Atlas and 19 times on an HPC system called Hera.

After more than a decade supporting a variety of applications and computing systems, SCR continues to evolve. Mohror states, "We want SCR's data management infrastructure to support even more complex workflows." As open-source software, the framework's influence can extend beyond the Laboratory. (See *S&TR*, January/February 2018, pp. 4–11.) Located at the University of California at San Diego, the San Diego Supercomputer Center is an example of an academic partner using SCR on production HPC applications. The application is useful for industry as well. Moody notes, "Our solution has worked so well that we want more people to benefit from it."

—Holly Auten



SCR leverages short-term storage for checkpoint files, thus accelerating data retrieval and subsequent restarting of the application if a computing failure occurs. The white dots represent data input/output movement between storage tiers and the scientific application. (Rendering by Ryan Goldsberry.)

Key Words: application programming interface, Argonne National Laboratory, checkpointing, high-performance computing (HPC), input/output (I/O) operation, open-source software, pF3D code, R&D 100 Award, Scalable Checkpoint/Restart (SCR) framework, scientific application, simulation.

For further information contact Kathryn Mohror (925) 423-2997 (mohror1@llnl.gov).