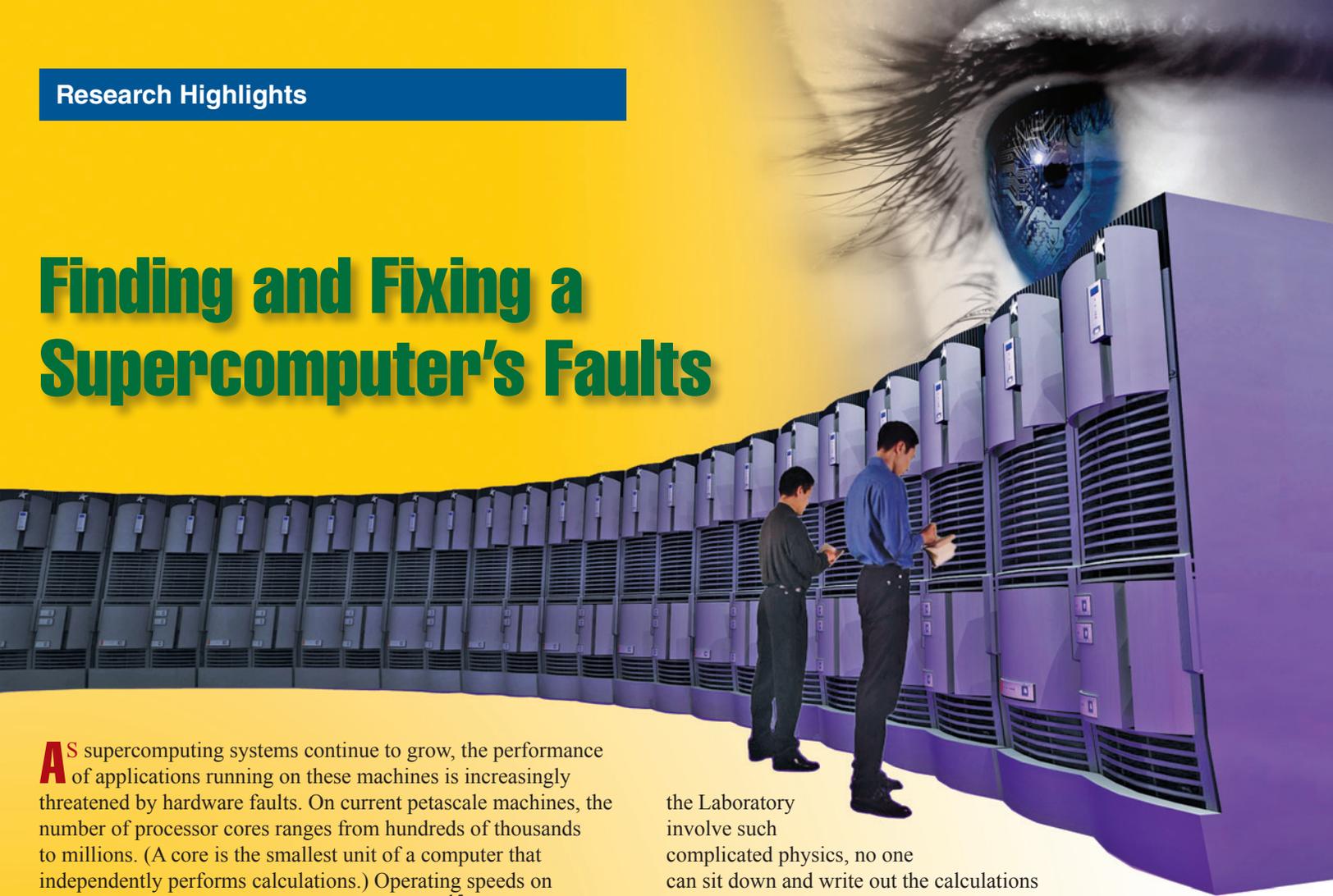


Finding and Fixing a Supercomputer's Faults



AS supercomputing systems continue to grow, the performance of applications running on these machines is increasingly threatened by hardware faults. On current petascale machines, the number of processor cores ranges from hundreds of thousands to millions. (A core is the smallest unit of a computer that independently performs calculations.) Operating speeds on petascale systems can exceed 1 quadrillion (10^{15}) floating-point operations per second (flops). By 2020, exascale systems made with hundreds of millions of cores will have 1,000 times the performance of today's petascale systems, running calculations at a rate of 1 quintillion (10^{18}) flops.

The large number of components on a supercomputing system increases the rate of hardware faults, which can cause applications to abort and performance to degrade. More importantly, they may corrupt results. To address this problem, computational scientists, with funding from the Laboratory Directed Research and Development Program, are developing methods to detect faults in supercomputers and help systems recover from errors that do occur, even on exascale systems.

Bronis de Supinski, who leads the Exascale Computing Technologies project in the Laboratory's Center for Applied Scientific Computing, says that the more nodes, components, and memory a system has, the greater its error rate will be. For example, he says, "If you're the only person driving down a road, there is a small chance you could have an accident. But if you're surrounded by 10,000 other cars, your chances of having an accident rise."

It is the same with computer systems. "We tend to think of computers as infallible," says de Supinski, "but physical processes—such as cosmic-ray strikes—can change the flow of electrons and affect what's on the memory cell."

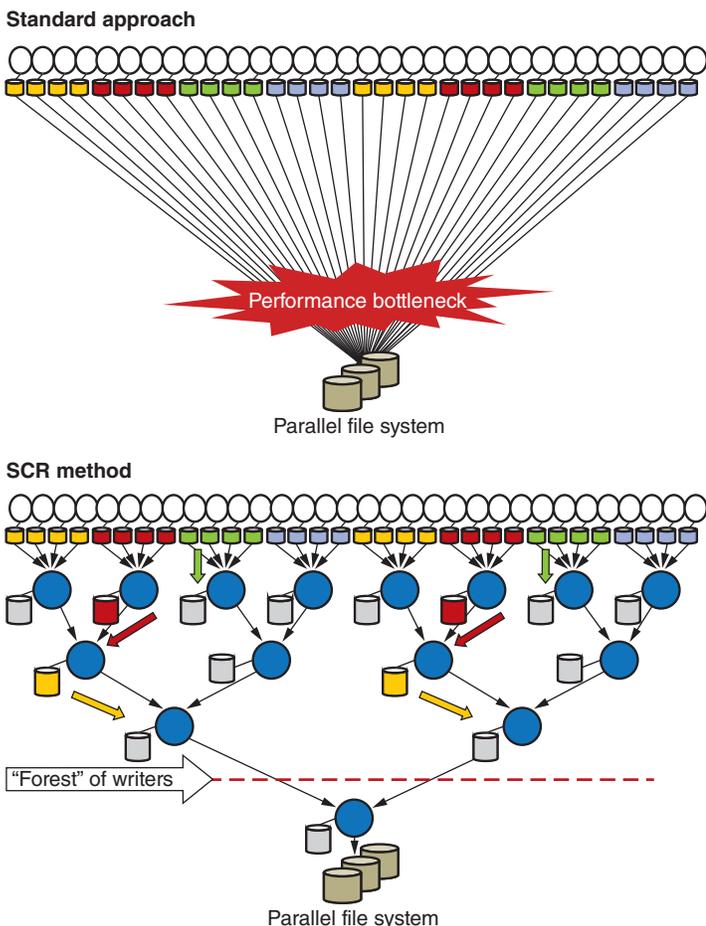
Computer scientist Greg Bronevetsky adds that a computer is a physical device, not an abstract idea. "The projects at

the Laboratory involve such complicated physics, no one can sit down and write out the calculations to solve them," says Bronevetsky, who received a Presidential Early Career Award for Scientists and Engineers in 2011. "The computer is a faster, much more powerful pencil, but like all devices made of parts from different vendors, an interaction between various components can lead to undesired outcomes. Something as simple as one chip dying—a problem we call a hard-stop fault—can cause the entire computation to crash."

Fault Finding

Hard-stop faults are the most common type of error and can stop an entire compute job. They are normally dealt with by writing checkpoints, a method in which the entire state of a job is saved and stored on a parallel file system. If a failure occurs, a program's state can be rolled back, or restored, from the most recent checkpoint, and operation resumed. But writing a single checkpoint to a parallel file system can require tens of minutes on a supercomputer. "Saving information somewhere else so we can bring it back grows expensive," de Supinski says. "If faults happen often, the system spends almost all of its time writing checkpoints and rolling back."

In addition, storage disk speeds are much slower than processor speeds. Their performance has not increased significantly over the years, even though processor performance has accelerated. Because of this bottleneck, an exascale system might spend more time saving and restoring information than performing computations.



The Scalable Checkpoint/Restart (SCR) method improves code performance by writing checkpoints to a compute node's local memory rather than to a parallel file system.

To reduce the time a machine requires to write checkpoints, a team of Livermore scientists led by Adam Moody developed the Scalable Checkpoint/Restart (SCR) approach. The SCR multilevel system can store checkpoints to a compute node's local memory—its random access or flash memory or even its disk—in addition to the parallel file system. Regular checkpoints can be saved quickly to local memory and duplicated on other nodes. If one node fails, its data can be restored from a duplicate. With this technique, the parallel file system is accessed much less frequently.

SCR stores, or caches, only the most recent checkpoints, discarding an older one as each new checkpoint is saved. It can also apply a redundancy scheme to the cache and recover checkpoints after a failure disables a small portion of the system. SCR proved its value when used with the pF3D code, which simulates laser-plasma interactions in support of the National Ignition Facility. In over 5 million node-hours of computation with pF3D, SCR

recovered 85 percent of the faults that occurred. "We are also looking into staggering the times at which checkpoints are written," de Supinski says. "Typically, a job computes for a while and then takes a checkpoint. Disk traffic dramatically peaks in bursts because many jobs are writing at once. We found that the higher demand on the file system—this burst of pounding—makes the system less reliable. SCR can help us smooth the input/output traffic."

The team extended SCR to use a technique called Remote Direct Memory Access, which pulls data off the node without involving the processor in data movement. Different nodes can be coordinated to schedule their writing to the file system. Moody and Livermore scientist Kathryn Mohror then worked with several summer students to compress multiple checkpoints into a single file and reduce the number of nodes writing to the file system at one time. This approach led to more reliable performance.

Flipping a Bit

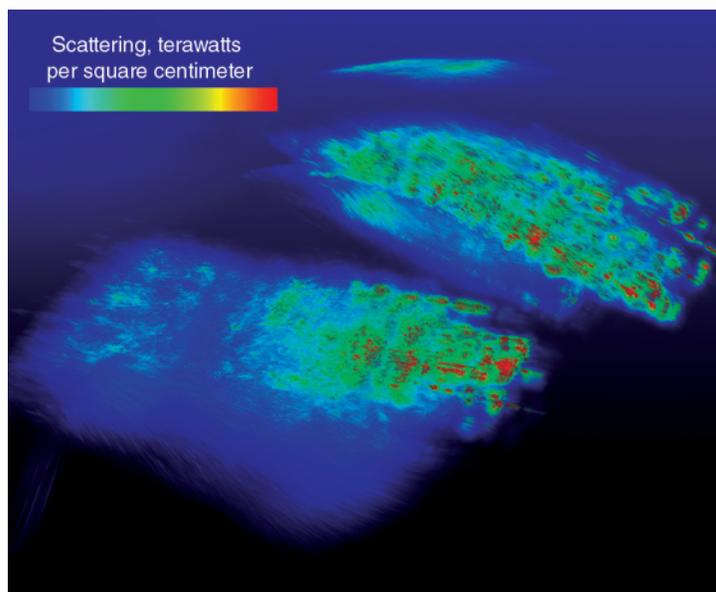
Another type of hardware error is a soft fault. These faults are insidious: Although the job continues to compute, the data are corrupted. "For example," says Bronevetsky, "if a charged particle goes through a transistor, it throws off a little piece of the computation." This error is known as "flipping a bit" because the binary code is switched from 0 to 1 or 1 to 0. When researchers analyzed BlueGene/L, the Laboratory's 108,000-node supercomputer, they found that one data-cache bit flip occurred every four hours. "The machines are expensive, and we want them to do as much productive work as possible," Bronevetsky says. "If an application that takes a week to run encounters a failure every four hours, the odds are that it will never complete."

To make applications more tolerant to bit flips, Livermore computer scientist and postdoctoral researcher Marc Casas Guix adapted the algebraic multigrid (AMG) algorithm, a powerful solver of sparse linear equations. AMG solves linear systems at multiple levels of granularity. The fine-grained solve steps reduce errors that result from inconsistencies between nearby grid cells, and the coarse-grained steps reduce inconsistencies between larger regions of space.

Laboratory researchers then flip bits in AMG to determine the code regions and data structures that are vulnerable to such errors. Based on the consequences, they can choose the most appropriate resilience strategy. To guard against hard failures, they can checkpoint data, automatically recalculate corrupted data structures, or run two copies of the same program simultaneously in case one version becomes corrupted. "AMG is good at overcoming errors by fixing them locally and iteratively," says Bronevetsky. "In practice, it survives faults well."

Clusters to the Rescue

When a supercomputer system computes correctly but the run time lasts much longer than normal, a performance fault is the culprit. Performance faults are challenging to identify because

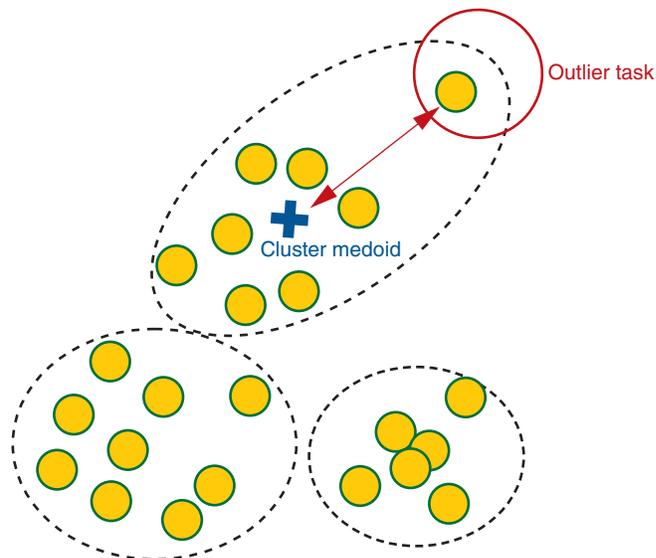


SCR recovered 85 percent of the faults that occurred in more than 5 million node-hours of computation with the pF3D code, which simulates laser-plasma interactions such as the one shown here.

they may occur on any of hundreds of thousands of processors in a petascale system. A Livermore team led by computer scientist Todd Gamblin has developed an automated machine-learning technique called CAPEK (Clustering Algorithm with Parallel Extended K-Medoids) that can quickly find faulty processors while a calculation is running.

CAPEK uses a fast sampling method that quickly identifies groups of processors with similar performance characteristics. This analysis gives each processor a general picture of the behavior of the system as a whole. “To know what’s abnormal, we must first determine what’s normal,” says Gamblin, who first studied load imbalance and developed compression techniques while collaborating with Livermore scientists on his dissertation. Once each processor receives a description of “normal” behavior from CAPEK, it can compare this to its own behavior and identify itself as either normal or faulty. Isolating the faulty processors significantly reduces the cost of analyzing an application’s performance. “CAPEK has proven to be a good method for determining when node behavior is different or suspicious,” says Gamblin. “With that information, we don’t have to examine all the nodes, only the ones affected.”

CAPEK is used for many types of analysis including statistical trace sampling and scalable detection of performance anomalies. Gamblin has worked with de Supinski, Bronevetsky, and collaborators at Purdue University to incorporate CAPEK with AutomaDeD, which automatically finds performance faults, so the tool can be scaled to next-generation systems. “Most traditional



A clustering algorithm called CAPEK (Clustering Algorithm with Parallel Extended K-Medoids) samples performance data from the various processors on a supercomputing system and locates small clusters with similar characteristics. The clusters quickly reveal outliers whose behavior is suspicious.

clustering algorithms get slower with large numbers of cores,” says Gamblin. “CAPEK doesn’t run much slower on 131,072 cores than it does on one. No matter how many cores are in use, it takes less than 1 second to run, which is fast enough for online use in production.”

Integrated Support for Laboratory Missions

De Supinski acknowledges that other new techniques are needed to keep predictive simulations running efficiently on exascale supercomputers. But SCR, the modified AMG algorithm, and CAPEK are important advances, making applications more tolerant of hardware faults.

“Ensuring the performance of our petascale and future exascale systems is critical to the success of many Laboratory missions,” says de Supinski. These machines provide the computational power researchers need for a wide range of 21st-century efforts, from modeling new materials to studying fusion reactions and predicting the effects of a changing climate.

—Kris Fury

Key Words: algebraic multigrid (AMG) algorithm, checkpoint, Clustering Algorithm with Parallel Extended K-Medoids (CAPEK), compute node, exascale computing, hard fault, performance fault, petascale computing, Scalable Checkpoint/Restart (SCR), soft fault, supercomputing.

For further information contact Bronis de Supinski (925) 422-1062 (bronis@llnl.gov).