

# Failure Tolerance in Petascale Computers

## Introduction

Three of the most difficult and growing problems in future high-performance computing (HPC) installations will be avoiding, coping and recovering from failures. The coming PetaFLOPS clusters will require the simultaneous use and control of hundreds of thousands or even millions of processing, storage, and networking elements. With this large number of elements involved, element failure will be frequent, making it increasingly difficult for applications to make forward progress. The success of petascale computing will depend on the ability to provide reliability and availability at scale.

While researchers and practitioners have spent decades investigating approaches for avoiding, coping and recovering from abstract models of computer failures, the progress in this area has been hindered by the lack of publicly available, detailed failure data from real large-scale systems.

We have collected and analyzed a number of large data sets on failures in high-performance computing (HPC) systems. Using these data sets and large scale trends and assumptions commonly applied to future computing systems design, we project onto the potential machines of the next decade our expectations for failure rates, mean time to application interruption, and the consequential application utilization of the full machine, based on checkpoint/restart fault tolerance and the balanced system design method of matching storage bandwidth and memory size to aggregate computing power.<sup>1</sup>

Not surprisingly, if the growth in aggregate computing power continues to outstrip the growth in per-chip computing power, more and more of the computer's resources may be spent on conventional fault recovery methods. For example, we envision applications being denied as much as half of the system's resources in five years.<sup>2</sup> The alternatives that might compensate for this unacceptable trend include application-level checkpoint compression, new special checkpoint devices or system level process-pairs fault-tolerance for supercomputing applications.

Our interest in large-scale cluster failure stems from our role in a larger effort, the DOE SciDAC-II Petascale Data Storage Institute (PDSI), chartered to anticipate and explore the challenges of storage systems for petascale computing.<sup>3</sup> In as much as checkpoint/restart is a driving application for petascale data storage systems, understanding node failure and application failure tolerance is an important function for the PDSI. To increase the benefit of our data collection efforts, and to inspire others to do the same, we are working with the USENIX Association to make publicly available these and other datasets in a Computer Failure Data Repository (CFDR).<sup>4</sup> Systems researchers and developers need to have ready access to raw data describing how computer failures have occurred on existing large-scale machines.

Garth Gibson  
Carnegie Mellon University

Bianca Schroeder  
Carnegie Mellon University

Joan Digney  
Carnegie Mellon University

---

<sup>1</sup> Grider, G. "HPC I/O and File System Issues and Perspectives," In *Presentation at ISW4, LA-UR-06-0473*, Slides available at [http://www.dtc.umn.edu/disc/isw/presentations/isw4\\_6.pdf](http://www.dtc.umn.edu/disc/isw/presentations/isw4_6.pdf), 2006.

<sup>2</sup> Schroeder, B., Gibson, G. "Understanding Failures in Petascale Computers," In *SciDAC 2007: Journal of Physics: Conference Series 78* (2007) 012022.

<sup>3</sup> Scientific Discovery through Advanced Computing (SciDAC), The Petascale Data Storage Institute (PDSI). <http://www.pdsi-scidac.org/>, 2006.

<sup>4</sup> The Computer Failure Data Repository (CFDR) - <http://cfd.usenix.org/>.

## Failure Tolerance in Petascale Computers

### Data Sources

The primary data set we are studying was collected between 1995 and 2005 at Los Alamos National Laboratory (LANL, [www.lanl.gov](http://www.lanl.gov)) and covers 22 high-performance computing systems, including a total of 4,750 machines and 24,101 processors.<sup>5</sup> Figure 1 shows pictures of two LANL systems. The data contain an entry for any failure that occurred during the nine year time period that resulted in an application interruption or a node outage. It covers all aspects of system failures: software failures, hardware failures, failures due to operator error, network failures, and failures due to environmental problems (e.g., power outages). For each failure, the data notes start time and end time, the system and node affected, as well as categorized root cause information. To the best of our knowledge, this is the largest failure data set studied to date, both in terms of the time-period it spans and the number of systems and processors it covers. It is also the first to be publicly available to researchers.<sup>6</sup>

<sup>5</sup> Schroeder, B., Gibson, G. "A large-scale study of failures in high-performance computing systems," In *Proc. of the 2006 International Conference on Dependable Systems and Networks (DSN'06)*, 2006.

<sup>6</sup> The LANL raw data and more information are available at: <http://www.lanl.gov/projects/computerscience/data/>.



Figure 1. Example high-performance computer clusters at Los Alamos National Laboratory, Blue Mountain (above) and ASC Q.

### Understanding Outages in LANL Computers

The first question most ask is “What causes a node outage?” Figure 2 provides a root cause breakdown of failures from the LANL data into human, environment, network, software, hardware, and unknown, with the relative frequency of the high-level root cause categories on the left. Hardware is the single largest source of malfunction, with more than 50% of all failures assigned to this category. Software is the second largest contributor, with around 20% of all failures. The trends are similar if we look at Figure 2(b), which shows the fraction of total repair time attributed to each of the different root cause categories.

## Failure Tolerance in Petascale Computers

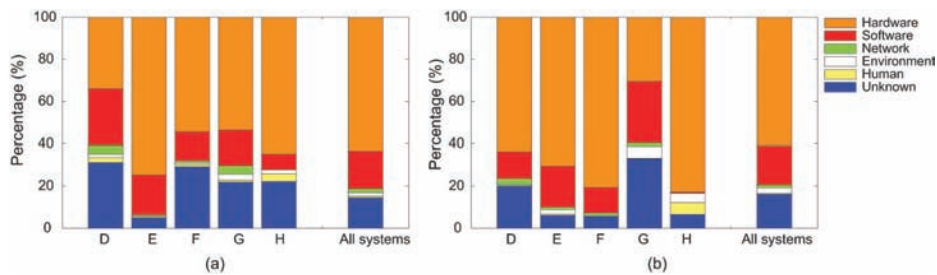


Figure 2. (a) The breakdown of failures by root cause. (b) The breakdown of total repair time spent on a system due to each root cause. Each bar shows the breakdown for the systems of one particular hardware platform, labeled D, E, F, G, and H, and the right-most bar shows aggregate statistics across all LANL systems.

It is important to note that the number of failures with undetermined root cause is significant. Since the fraction of hardware failures is larger than the fraction of undetermined failures, and the fraction of software failures is close to that of undetermined failures, we can still conclude that hardware and software are among the largest contributors to failures. However, we cannot conclude that any of the other failure sources (Human, Environment, Network) is actually insignificant.

A second question is “How frequently do node outages occur?” or “How long can an application be expected to run before it will be interrupted by a node failure?” Figure 3(a) shows the average number of node failures observed per year for each of the LANL systems according to the year that each system was introduced into use. The figure indicates that the failure rates vary widely across systems, from less than 20 failures per year per system to more than 1100 failures per year. Note that a failure rate of 1100 per year means that an application running on all the nodes of the system will be interrupted and forced into recovery more than two times per day. Since many of the applications running on these systems require a large number of nodes and weeks of computation to complete, failure and recovery are frequent events during an application’s execution.

One might wonder what causes the large differences in failure rates across the different systems. The main reason for these differences is that the systems vary widely in size. Figure 3(b) shows the average number of failures per year for each system normalized by the number of processors in the system. The normalized failure rates show significantly less variability across the different types of systems, which leads us to two interesting suggestions. *First, the failure rate of a system grows in proportion to the number of processor chips in the system. Second, there is little indication that systems and their hardware get more reliable over time as technology changes.*

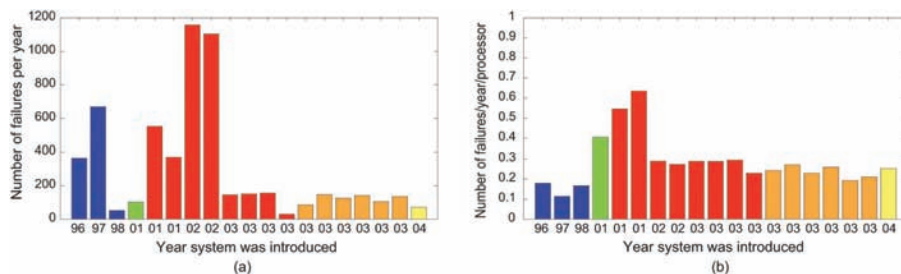


Figure 3. (a) Average number of failures for each LANL system per year. (b) Average number of failures for each system per year normalized by number of processors in the system. Systems with the same hardware type have the same color.

## Failure Tolerance in Petascale Computers

### Lower Mean Time To Interrupt (MTTI) in Petascale Computers

What does our data analysis, examined in the light of recent technology trends, predict for the reliability and availability of future HPC systems?

Our essential prediction is that the number of processor chips will grow with time, increasing failure rates and fault tolerance overheads.

First, we expect petascale computers will be conceived and constructed according to long standing trends (aggregate compute performance doubling every year) shown on the top500.org list of the largest documented computers.<sup>7</sup> Second, we expect little or no increase in clock speed, but an increase in the number of processor cores per processor chip, commonly referred to as a socket in the new multi-core processor era, at a fast rate, estimated as doubling every two years.<sup>8</sup> Our data also predicts that failure rates will grow in proportion to the number of sockets in the system and that there is no indication that the failure rate per socket will decrease over time with technology changes. Therefore, as the number of sockets in future systems increases to achieve top500.org performance trends, we expect the system wide failure rate will increase.

In an attempt to quantify what one might expect to see in future systems, we examined the LANL data and found that an optimistic estimate for the failure rate per year per socket is 0.1. Our data does not predict how failure rates will change with increasing numbers of cores per processor chip core, but it is reasonable to predict that many failure prone mechanisms operate at the chip level, so we make the (possibly highly optimistic) assumption that failure rates will increase only with the number of chip sockets, and not with the number of cores per chip.

As a baseline for our projections, we modeled the Jaguar system at Oak Ridge National Laboratory (ORNL). After it is expanded to a Petaflop system in 2008, Jaguar is expected to have around 11,000 processor sockets (dual-core Opteron), 45 TB of main memory and a storage bandwidth of 55 GB/s.<sup>9</sup> Predictions for system expansion are bracketed with three projected rates of growth, with numbers of cores doubling every 18, 24 and 30 months.

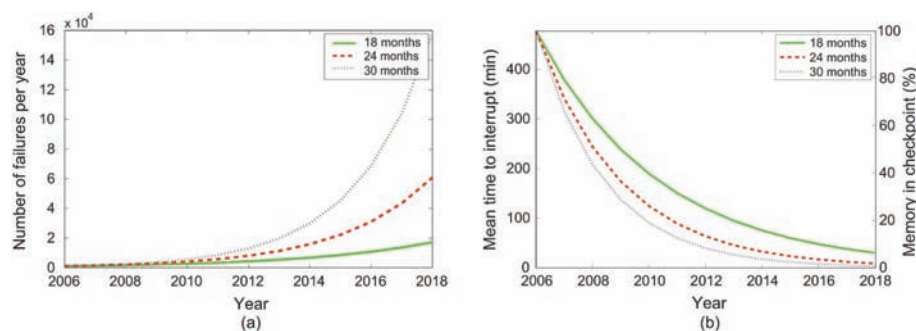


Figure 4. (a) The expected growth in failure rate and (b) decrease in MTTI, assuming that the number of cores per socket grows by a factor of two every 18, 24 and 30 months, respectively, and the number of sockets increases so that aggregate performance conforms to top500.org.

Figure 4 plots the expected increase in failure rate and corresponding decrease in mean time to interrupt (MTTI), based on the above assumptions. Even if we assume a zero increase in failure rate with more cores per socket (a stretch), the failure rates across the biggest machines in the top 500 lists of the future can be expected to grow dramatically.

<sup>7</sup> Top 500 supercomputing sites - <http://www.top500.org/>, 2007.

<sup>8</sup> Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W., Yelick, K. A. "The landscape of parallel computing research: A view from Berkeley," *Technical Report UCB/EECS-2006-183*, EECS Department, University of California, Berkeley, Dec. 2006.

<sup>9</sup> Roth, P. C. "The Path to Petascale at Oak Ridge National Laboratory," In *Petascale Data Storage Workshop Supercomputing '06*, 2006.

## Failure Tolerance in Petascale Computers

### Decreasing Effectiveness of Checkpoint-Restart Fault Tolerance

Observing this sort of dramatic increase in failure rates brings up the question of how the utility of future systems will be affected. Fault tolerance in HPC systems is typically implemented with checkpoint restart programming. Here, the application periodically stops useful work to write a checkpoint to disk. In case of a node failure, the application is restarted from the most recent checkpoint and recomputes the lost results.

The time to write a checkpoint depends on the total amount of memory in the system, the fraction of memory the application needs to checkpoint to be able to recover, and the I/O bandwidth. To be conservative, we assume that demanding applications may utilize and checkpoint their entire memory. For a system like Jaguar, with 45TB of memory and 55 GB/s of storage bandwidth, that means one system-wide checkpoint will take on the order of 13 minutes. In a balanced system model, where bandwidth and memory both grow in proportion to compute power, the time to write a checkpoint will stay constant over time. However, with failures becoming more frequent, restarting will be more frequent and application work will be recomputed more frequently. Reducing the time between checkpoints reduces the amount of work recomputed on a restart but it also increases the fraction of each checkpoint interval spent taking a checkpoint.

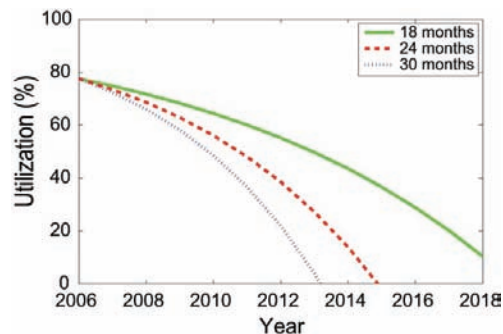


Figure 5. Effective application utilization drops because mean time to interrupt is dropping and more time will be lost to taking checkpoints and restarting from checkpoints. The three models are the same as in Figure 4.

Based on the models of Figure 4 and on an optimal selection of the period between checkpoints,<sup>10</sup> Figure 5 shows a prediction that the effective resource utilization by an application will drastically decrease over time. For example, in the case where the number of cores per chip doubles every 30 months, the utilization drops to zero by 2013, meaning the system is spending 100% of its time writing checkpoints or recovering lost work, a situation that is clearly unacceptable. In the next section we consider possible ways to stave off this projected drop in resources utilization.

<sup>10</sup>Young, J. W. "A first order approximation to the optimum checkpoint interval," *Commun. ACM*, 17(9):530–531, 1974.

### Better Fault Tolerance for Petascale Computers

As LANL's data suggests that failure rate grows proportionally to the number of sockets, keeping the number of sockets constant should stave off an increase in the failure rate. To do this, however, means either failing to achieve the top500.org aggregate performance trends, or increasing the performance of each processor chip faster than currently projected.<sup>8</sup> Chip designers consider it unlikely that we will see a return to the era of rapid decreases in processor cycle time because of the power consumption implications. The remaining alternative, increasing the number of cores per chip faster, would probably not be effective, even if it was possible, because memory bandwidth

## Failure Tolerance in Petascale Computers

per chip will not keep up. Therefore, we think the number of processor chip sockets will continue to increase to keep performance on the top500.org trends.

*Socket Reliability:* The increase in failure rates could also be prevented if individual processor chip sockets were made more reliable each year in the future, i.e., if the per socket MTTI would increase proportionally to the number of sockets per system over time. Unfortunately, LANL's data does not indicate that hardware has become more reliable over time, suggesting that as long as the number of sockets is rising, the system-wide MTTI will drop.

*Partitioning:* The number of interrupts an application sees depends on the number of processor chips it is using in parallel. One way to stave off the drop in MTTI per application would be to run it only on a constant-sized sub-partition of the machine, rather than on all nodes of a machine. Unfortunately, while this solution works for small applications that do not need performance to increase faster than the speed each chip increases, it is not appealing for the most demanding "hero" applications, for which the largest new computers are often justified.

*Faster Checkpointing:* The basic premise of the checkpoint restart approach to fault tolerance, often called the balanced system design, is that even if MTTI is not decreasing, storage bandwidth increases in proportion to total performance.<sup>1</sup> Though achieving balance (i.e., doubling of storage bandwidth every year) is a difficult challenge for storage systems, one way to cope with increasing failure rates is to effect further increases in storage bandwidth. For example, assuming that the number of sockets and hence the failure rate grows by 40% per year, the effective application utilization would stay the same if checkpoints were taken in 30% less time each year. Projections show that this sort of increase in bandwidth is orders of magnitude higher than the commonly expected increase in bandwidth per disk drive (generally about 20% per year). Therefore, an increase in bandwidth would have to come from a rapid growth in the total number of drives, well over 100% per year, increasing the cost of the storage system much faster than any other part of petascale computers. This might be possible, but it is not very desirable.

Another option is to decrease the amount of memory being checkpointed, either by not growing total memory as fast, or by better compression of application data leading to only a smaller fraction of memory being written in each checkpoint. Growing total memory at a slower than balanced rate will help reduce total system cost, which is perhaps independently likely, but may not be acceptable for the most demanding applications. Of the two, compression seems to be the more appealing approach, and is entirely under the control of application programmers.

Achieving higher checkpoint speedups purely by compression will require significantly better compression ratios each year. As early as in the year 2010, an application will have to construct its checkpoints with a size at most 50% of the total memory. Once the 50% mark is crossed, other options, such as diskless checkpointing where the checkpoint is written to the volatile memory of another node rather than disk,<sup>11</sup><sup>12</sup> or hybrid approaches<sup>13</sup> become viable. We recommend that any application capable of compressing its checkpoint size should pursue this path; considering the increasing number of cycles that will go into checkpointing, the compute time needed for compression may be time well spent.

A third approach to taking checkpoints faster is to introduce special devices between storage and memory that will accept a checkpoint at speeds that scale with memory, then relay the checkpoint to storage after the application has resumed computing. Although such an intermediate memory could be very expensive as it is as large as

<sup>11</sup> Plank, J. S., Li, K. "Faster checkpointing with N + 1 parity," In *Proc. 24th International Symposium on Fault Tolerant Computing*, 1994.

<sup>12</sup> Plank, J. S., Li, K., Puening, M. A. "Diskless checkpointing," *IEEE Trans. Parallel Distrib. Syst.*, 9(10):972–986, 1998.

<sup>13</sup> Vaidya, N. H. "A case for two-level distributed recovery schemes," In *Proceedings of the 1995 ACM SIGMETRICS conference*, 1995.

## Failure Tolerance in Petascale Computers


memory, it might be a good application for cheaper but write-limited technologies such as flash memory, because checkpoints are written infrequently.

*Non-Checkpoint-based Fault Tolerance:* Process-pairs duplication and checking of all computations is a traditional method for tolerating detectable faults that hasn't been applied to HPC systems.<sup>14,15,16</sup> Basically, every operation is done twice in different nodes so the later failure of a node does not destroy the operation's results. Process pairs would eliminate both the cost associated with writing checkpoints, because they are not needed, as well as lost work in the case of failure. However, using process pairs is expensive in that it requires giving up 50% of the hardware to compute each operation twice in different nodes and it introduces further overheads to keep process pairs in synch. However, if no other method works to keep utilization above 50%, this sacrifice might become appropriate, and it bounds the decrease in effectiveness to about 50%, perhaps without requiring special hardware.

## Conclusions

The most demanding applications, often the same applications that justify the largest computers, will see ever-increasing failure rates if the trends seen at top500.org continue. Using the standard checkpoint restart fault tolerance strategy, the efficacy of petascale machines running demanding applications will fall off. Relying on computer vendors to counter this trend is not recommended by historical data, and relying on disk storage bandwidth to counter it is likely to be expensive at best. We recommend that these applications consider spending an increasing number of cycles compressing checkpoints. We also recommend experimentation with process pairs fault tolerance for supercomputing. And if technologies such as flash memory are appropriate, we recommend experimenting with special devices devoted to checkpointing.

## The Computer Failure Data Repository

The work described in this article is part of our broader research agenda with the goal of analyzing and making publicly available the failure data from a large variety of real production systems. To date, large-scale studies of failures in real production systems are scarce, probably a result of the reluctance of the owners of such systems to release failure data. Thus, we have built a public *Computer Failure Data Repository* (CFDR), hosted by the USENIX association<sup>4</sup> with the goal of accelerating research on system reliability by filling the nearly empty collection of public data with detailed failure data from a variety of large production systems. We encourage all petascale computing organizations to collect and publish failure data for their systems in the repository. 

### Acknowledgments

We would like to thank Jamez Nunez and Gary Grider from the High Performance Computing Division at Los Alamos National Lab for collecting and providing us with data and helping us to interpret the data. We thank the members and companies of the PDL Consortium (including APC, Cisco, Google, EMC, Hewlett-Packard, Hitachi, IBM, Intel, LSI, Microsoft, Network Appliance, Oracle, Panasas, Seagate, and Symantec) for their interest and support. This material is based upon work supported by the Department of Energy under Award Number DE-FC02-06ER25767 [3] and on research sponsored in part by the Army Research Office, under agreement number DAAD19-02-1-0389.

<sup>14</sup> Bressoud, T. C., Schneider, F. B., "Hypervisor-based fault tolerance," *ACM Trans. Comput. Syst.*, 14(1):80–107, 1996.

<sup>15</sup> Chapin, J., Rosenblum, M., Devine, S., Lahiri, T., Teodosiu, D., Gupta, A. "Hive: fault containment for shared-memory multiprocessors," In *SOSP '95: Proceedings of the fifteenth ACM symposium on Operating systems principles*, 1995.

<sup>16</sup> McEvoy, D. "The architecture of tandem's nonstop system," In *ACM 81: Proceedings of the ACM '81 conference*, page 245, New York, NY, USA, 1981. ACM Press.