Understanding Failures in Petascale Computers

Bianca Schroeder **Garth A. Gibson**

Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA

E-mail: bianca@cs.cmu.edu, garth@cs.cmu.edu

Abstract.
With petascale computers only a year or two away there is a pressing need to anticipate and compensate for a probable increase in failure and application interruption rates. Researchers, designers and integrators have available to them far too little detailed information on the failures and interruptions that even smaller terascale computers experience. The information that is available suggests that application interruptions will become far more common in the coming decade, and the largest applications may surrender large fractions of the computer's resources to taking checkpoints and restarting from a checkpoint after an interruption. This paper reviews sources of failure information for compute clusters and storage systems, projects failure rates and the corresponding decrease in application effectiveness, and discusses coping strategies such as application-level checkpoint compression and system level process-pairs fault-tolerance for supercomputing. The need for a public repository for detailed failure and interruption records is particularly concerning, as projections from one architectural family of machines to another are widely disputed. To this end, this paper introduces the Computer Failure Data Repository and issues a call for failure history data to publish in it.

1. Introduction

One of the hardest 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 failures, the progress in this area has been hindered by the lack of publicly available 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. These data sets cover node outages in HPC clusters, as well as failures in storage 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 [14].

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. We envision applications being denied as much as half of the system's resources in five years, for example. We then discuss alternative actions that may compensate for this unacceptable

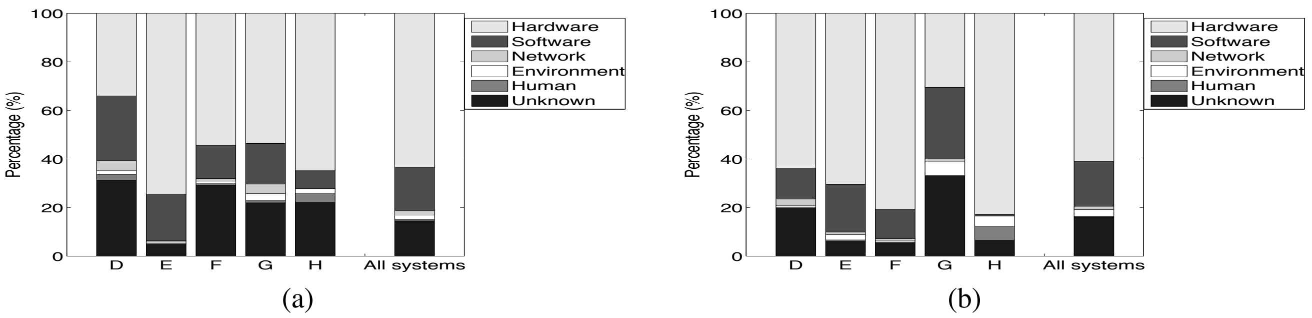

Figure 1. *The breakdown of failures by root cause. Each bar shows the breakdown for the systems of one particular hardware platform and the right-most bar shows aggregate statistics across all LANL systems.*

trend. We also briefly discuss our efforts to make publicly available much of these data sets in a *Computer Failure Data Repository (CFDR)* to be hosted by USENIX. With the increasing importance of frequent failures during petascale application execution, we assert that computer application designers need ready access to raw data describing how computer failures have occurred on existing machines.

2. Data sources

2.1. Node outages in HPC clusters

The primary data set we obtained was collected during 1995–2005 at Los Alamos National Laboratory (LANL) and covers 22 high-performance computing systems, including a total of 4,750 machines and 24,101 processors. 18 of the systems are SMP-based clusters with 2 to 4 processors per node, comprising a total of 4,672 nodes and 15,101 processors. The four remaining clusters consist of NUMA boxes with 128 to 256 processors each, adding up to a total of 78 nodes and 9,000 processors. The data contain an entry for any failure that occurred during the 9- year time period and that resulted in an application interruption or a node outage. The data 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 includes 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 in the literature to date, both in terms of the time-period it spans, and the number of systems and processors it covers, and the first to be publicly available to researchers (see [2] for raw data). In Section 3, we provide a few of our results from analyzing this data [30].

2.2. Storage failures

Our interest in large-scale cluster failure originates in the key role of high bandwidth storage in checkpoint/restart strategies for application fault tolerance [11]. We are part of a larger effort, the DOE SciDAC-II Petascale Data Storage Institute, chartered to anticipate and explore the challenges of storage systems for petascale computing [4]. Although storage failures are often masked from interrupting applications by RAID technology [24], reconstructing a failed disk can impact storage performance noticeably and if too many failures occur, storage system recovery tools can take days to bring a large filesystem back online, perhaps without all of its user's precious data. Moreover, disks have traditionally been viewed as perhaps the least reliable hardware component, due to the mechanical aspects of a disk.

We have been able to obtain four data sets (referred to as HPC1 – HPC4) describing disk drive failures occurring at HPC sites and three data sets (referred to as COM1 – COM3) collected at a large internet service provider. The data sets vary in duration from 1 month to 5 years and cover a total of more than 100,000 hard drives from four different vendors, and include SCSI, fibre-channel and SATA disk drives. We provide a few of our results from analyzing the data in Section 4. For more detailed results see [31].

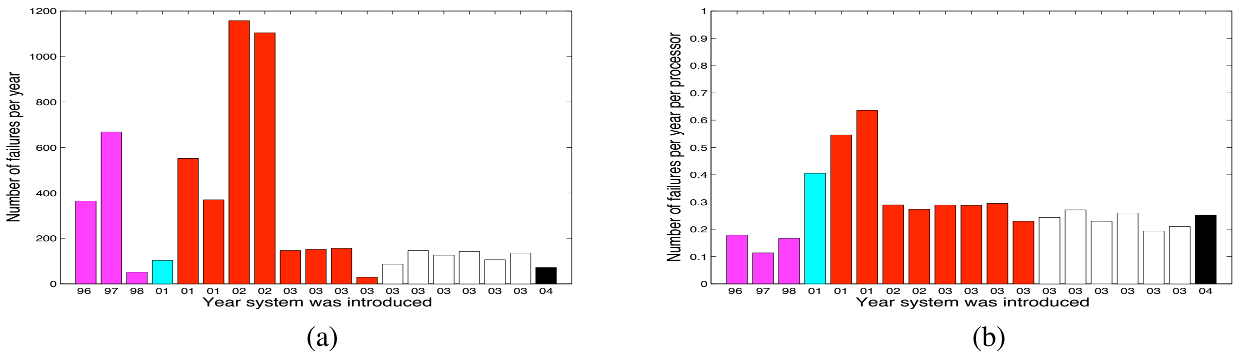

Figure 2. *(a) Average number of failures for each LANL system per year. (b) Average number of failures for each system per year normalized by number of processors in the system. Systems with the same hardware type have the same color.*

3. Cluster node outages

3.1. What's the most common root cause of node outages?

The first question most ask is what causes a node outage. The LANL data provides a root cause breakdown of failures into human, environment, network, software, hardware, and unknown. Figure 1 (left) shows the relative frequency of the high-level root cause categories. Hardware is the single largest component, with more than 50% of all failures assigned to this category. Software is the second largest contributor, with around 20% of all failures at both sites attributed to software. The trends are similar if we look at Figure 1 (right), the fraction of total repair time attributed to each of the different root cause categories.

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 can not conclude that any of the other failure sources (Human, Environment, Network) is insignificant.

3.2. How often do nodes fail?

Another question is how frequently node outages occur, or in other words, how long can you expect an application to run before it will be interrupted by a node failure. Figure 2 (left) shows the average number of node failures observed per year for each of the LANL systems. Systems that are based on the same hardware technology (same chip make and model) are shaded in the same color. The graph indicates that the failure rate varies 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 failures 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. It turns out that the main reason for these differences is that the systems vary widely in size. Figure 2(b) shows for each system the average number of failures per year normalized by the number of processors (sockets) in the system. The normalized failure rates show significantly less variability across systems, which leads us to two interesting propositions. *First, the failure rate of a system grows proportional 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 2 shows the systems sorted by the time they went into production and the normalized failure rates exhibit neither an increasing nor a decreasing trend moving from the oldest system to the most recent system.

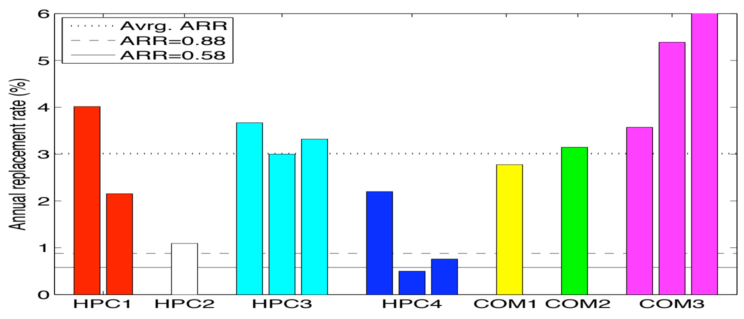

Figure 3. *Comparison of datasheet AFRs (solid and dashed line in the graph) and ARRs observed in the field. Each bar in the graph corresponds to one drive population. The dotted line represents the weighted average over all data sets.*

4. Storage failures

4.1. How often do drives need to be replaced?

A common way to estimate the rate of drive replacement is to use the mean-time-to-failure (MTTF), or its inverse, the annual failure rate (AFR), specified in the drive's datasheet provided by the vendor. For modern drives, the datasheet MTTFs are typically in the range of 1-1.5 million hours, suggesting an annual failure and replacement rate (ARR) between 0.58% to 0.88%. In the data, however, field experience with disk replacements differs from datasheet specifications of disk reliability. Figure 3 shows the annual failure rate suggested by the datasheets (horizontal solid and dashed line), the observed ARRs for each of the seven data sets and the weighted average ARR for all disks less than five years old (dotted line).

For HPC1, HPC3, HPC4 and COM3, which cover different types of disks, the graph contains several bars, one for each type of disk. Figure 3 shows a significant discrepancy between the observed ARR and the datasheet AFR for all data sets. While the datasheet AFRs are between 0.58% and 0.88 from 0.5% to as high as 13.5%. That is, the observed ARRs by data set and type, are by up to a factor of 15 higher than datasheet AFRs. The average ARR over all data sets (weighted by the number of drives in each data set) is 3.01%. Even after removing all COM3 data, which exhibits the highest ARRs, the average ARR was still 2.86%, 3.3 times higher than 0.88%.

A natural question arises: why are the observed disk replacement rates so much higher in the field than the datasheet MTTF would suggest, even for drives in the first years of operation. In discussions with vendors and customers, we have identified a number of possible reasons. First, customers and vendors might not always agree on the definition of when a drive is “faulty”. The fact that a disk was replaced implies that it failed some (possibly customer specific) health test. When a health test is conservative, it might lead to replacing a drive that the vendor tests would find to be healthy. Second, datasheet MTTFs are typically determined based on accelerated (stress) tests, which make certain assumptions about the operating conditions under which the disks will be used (e.g. that the temperature will always stay below some threshold), the workloads and “duty cycles” or powered-on hours patterns, and that certain data center handling procedures are followed. In practice, operating conditions might not always be as ideal as assumed in the tests used to determine datasheet MTTFs.

4.2. How does drive age affect replacement rates?

Disk replacement rates are commonly assumed to vary over a system's life cycle as drives age. The first year of drive operation is commonly characterized by early failures (or infant mortality). In years 2-5, the failure rates are expected to be approximately in steady state, and then, after years 5-7, wear-out starts to kick in and as a result failure rates increase [9, 10, 38].

We used data sets from three drive populations (the drives in the compute nodes and filesystem nodes of HPC1 and the first type of HPC4 drives) to study the failure rate pattern as a function of age. We make two interesting observations. First, replacement rates are rising significantly over the years, even during early years in the lifecycle. Replacement rates in HPC1 nearly double from year 1 to 2, or from year 2 to 3. This observation suggests that wear-out may start much earlier than expected, leading to steadily increasing replacement rates during most of a system's useful life, rather than steady failure rates during most of a drive's lifetime.

Second, we observe hardly any infant mortality. For the HPC1 file system nodes and for the HPC4 drives, the ARR of drives is not higher in the first few months of the first year than the last few months of the first year. In the case of the HPC1 compute nodes, infant mortality is limited to the first month of operation and is not above the steady state estimate of the datasheet MTTF.

5. Projections for petascale computers

This section looks at recent technology trends and combines them with results from our data analysis to provide some projections on the reliability and availability of future HPC systems.

5.1. Projected failure rates model

Our essential prediction is that the number of processor chips will be growing with time, increasing failure rates and fault tolerance overheads. We argue as follows. First, we expect integrators to deliver petascale computers according to the long standing trends shown on top500.org [5]; that is, the aggregate compute performance doubles every year. Second, we expect future processor chip designers to provide little or no increase in clock speed, but to increase the number of processor cores per processor chip, commonly referred to as a socket in the new many-core processor era, at a fast rate, estimated as doubling every two years [6]. In the models that follow we bracket this prediction with three rates of growth in cores: doubling every 18, 24 and 30 months. Even if HPC programmers can manage to increase thread parallelism at this impressive rate, achieving top500.org trends will require more processor chip sockets per system each year, at a growth rate of a factor of 1.26, 1.4 or 1.51 every year, respectively.

We have proposed that the data in Section 2 predicts failure rates will grow proportional 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.

The following projections attempt to quantify the increase in failure rate one might expect to see in future systems. First, based on the LANL data show in Figure 2, 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 mechanisms operate at the chip level, so we make the possibly highly optimistic assumption that failure rates increase only with number of chip sockets, and not with the number of cores per chip.

As a baseline for our projections, we model the Jaguar system at Oak Ridge National Laboratory (ORNL) after it is expanded to a Petaflop system in 2008. Jaguar will then have around 11,000 processor sockets (dual-core Opterons), 45 TB of main memory and a storage bandwidth of 55 GB/s [28]. While the architecture of other 2008 petascale machines, such as LANL's Roadrunner [17], differs from Jaguar in its use of hybrid nodes employing vector/graphics co- processors, our predictions for its failure rates are little different from Jaguar, so we do not include separate lines for it on our graphs[1].

Figure 4 plots the expected increase in failure rate and corresponding decrease in mean time to interrupt (MTTI), based on the above assumptions. The graphs shows that even if we grant zero increase in failure rate with more cores per socket (a stretch), the failure rates across peak machines in the top 500 curves can be expected to grow dramatically in the future.

[1] There is some suggestion that Blue Gene machines experience lower failure rates [33], but we have no data on which to make an argument for a dramatically different projection for Blue Gene computers

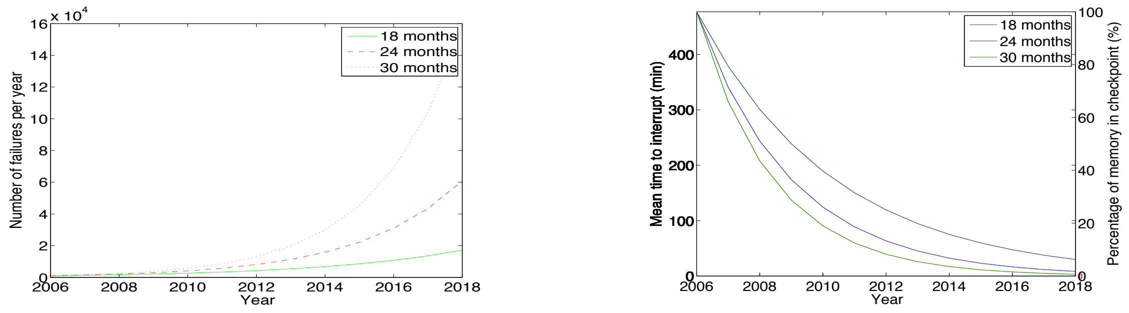

Figure 4. The expected growth in failure rate (left) and decrease in MTTI (right), assuming that the number of cores per socket grows by a factor of 2 every 18, 24 and 30 months, respectively, and the number of sockets increases to stay on top500.org.

5.2. Modeling checkpoint overhead

Observing the dramatic increase in failure rates in Figure 4 brings up the question of how this increase in failure rates will affect the utility of future systems. Fault tolerance in HPC systems is typically implemented with checkpoint restart programming. With checkpoint restart programs, the applications 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 effective application utilization of a system is the amount of time it is doing useful work, i.e. executing user code, rather than writing checkpoints or redoing work that has been lost after a node failure. The utilization will depend on the time t it takes to write a checkpoint, the mean time between interrupt $MTTI$ and the time between checkpoints i, and can be computed as

$$1 - (\frac{1}{2} \cdot i \cdot \frac{1}{MTTI} + t \cdot \frac{1}{i}) \tag{1}$$

The time t to write a checkpoint depends on the total amount of memory m in the system, the fraction f of memory the application needs to checkpoint to be able to recover, and the I/O bandwidth b, specifically, $t = m \cdot f/b$. To be conservative we assume that demanding applications may utilize and checkpoint the entire memory ($f = 1$). 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. For future projections, we assume a balanced system model, where bandwidth and memory both grow in proportion to compute power, and hence the time t to write a checkpoint will stay constant over time. The remaining variable in Equation 1 is the checkpoint interval i. To balance the trade-off between the amount of lost work (minimized with small i) versus the total time the system spends writing checkpoints (minimized with large i), the checkpoint interval i is often chosen as $\sqrt{2 \cdot MTTI \cdot t}$ [39].

Figure 5 shows a projection of effective application utilization over time, based on the above assumptions. We observe that utilization is drastically decreasing 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 would spend 100% of its time writing checkpoints or recovering lost work, a situation that is clearly unacceptable. In the remainder of this section we consider different strategies to stave off this predicted drop in utilization.

5.3. No grow in number of sockets per system

As our data suggests that failure rate grows proportional to the number of sockets, keeping the number of sockets constant would stave off an increase in the failure rate. To do this, however, means either failing to achieve the top500.org aggregate performance trends, an outcome that the HPC community appears determined to avoid, or increasing the performance of each processor chip faster than currently

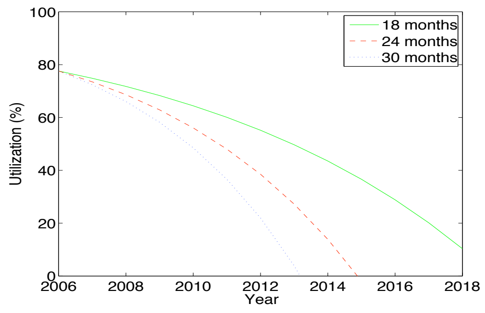

Figure 5. Effective application utilization over time.

projected [6]. 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 per chip will not keep up. Therefore, we think the number of processor chip sockets will increase continually.

5.4. Make individual sockets more reliable

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. While chip designers offer no expectation of this happening, it does seem like it is the favorite choice of system integrators in that each new request for proposed machines seems to hold the system MTTI constant. Unfortunately, our data (recall Figure 2) indicates that hardware has not become more reliable over time, suggesting that as long as the number of sockets is rising, the system-wide MTTI will drop.

5.5. Partition systems

The number of interrupts an application sees depends on the number of processor chips it is using in parallel. One way to increase the MTTI seen by an application would be to run it only on a sub-partition of the machine, rather than on all nodes of a machine. Revisiting the equations in the beginning of this section reveals that this approach is actually effective: e.g. cutting the number of nodes an application runs on in half will also cut the amount of work lost due to interrupts and due to writing checkpoints in half. Another way to view this approach is to use many smaller machines, rather than one big machine. While this solution works for small applications, it is not appealing for the most demanding, ”hero” applications for which the largest new computers are often justified. We therefore think, that while this approach will be used extensively, it does not solve the problem for the most demanding applications.

5.6. Take checkpoints faster

The basic premise of the checkpoint restart approach to fault tolerance, even if MTTI is not decreasing, is that storage bandwidth increases in proportion to total performance, often called the balanced system design [14]. Even though achieving balance, meaning a doubling of storage bandwidth every year, is already a difficult challenge for storage systems, one way to cope with increasing failure rates is to further increase 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. Figure 6 shows the increase in storage bandwidth necessary to keep up with failure rates, assuming that the number of cores doubles every 18, 24 and 30 months, respectively. Note that the required increase in bandwidth is orders of magnitude higher than the expected increase in bandwidth per disk drive (commonly assumed to be 20% per year and shown in the bottom line in

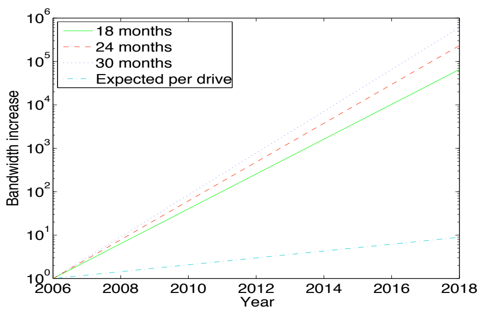

Figure 6. The growth in bandwidth necessary to make checkpoints cheap enough to compensate for the increased failure rate due to growth in number of sockets, assuming the number of cores per socket grows by a factor of 2 every 18, 24 and 30 months, respectively.

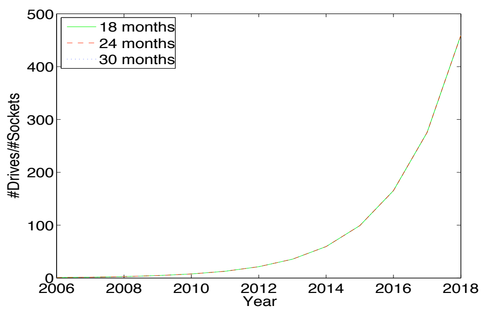

Figure 7. The ratio of number of disks to number of sockets when growing disk bandwidth to compensate for the increased failure rate due to growth in number of sockets.

Figure 6). This means the increase in bandwidth will have to come from a growth in the total number of drives. This growth will have to be significantly higher than the expected growth in the number of sockets. Figure 7 shows the ratio of number of drives to number of chip sockets to provide the necessary bandwidth. Treating the price of a chip and the price of a drive as fixed, reasonable for business goals, Figure 7 implies that both the fraction of cost going into storage and the total system cost will have to increase significantly in the future. This might be possible, but it is not very desirable.

A second 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 fraction of 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. Compression seems to be the more appealing approach.

Achieving higher checkpoint speedups purely by compression will require significantly better compression ratios each year. Figure 4 (right) shows, on the right Y-axis, the maximum fraction of (balance-sized) memory an application can write back in each checkpoint in order to provide the necessary speedup to compensate for increasing failure rates. Note that as early as in the year 2010 an application would have to checkpoint 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 [26, 27], or hybrid approaches [35] 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.

5.7. Use process pairs for fault tolerance rather than checkpoint restart

One might also consider alternative techniques to the currently standard checkpoint/restart methods for fault tolerance. One traditional method for fault tolerance that hasn't been applied to HPC systems yet is the use of process pairs checking each other on all computations [7, 8, 19]. Process pairs would eliminate both the cost associated with writing back checkpoints, as well as lost work in the case of failure.

However, using process pairs is expensive in that it requires giving up 50% of hardware and introduces overheads to keep processors checking. However, once no other method works to keep up utilization, this sacrifice might become appropriate.

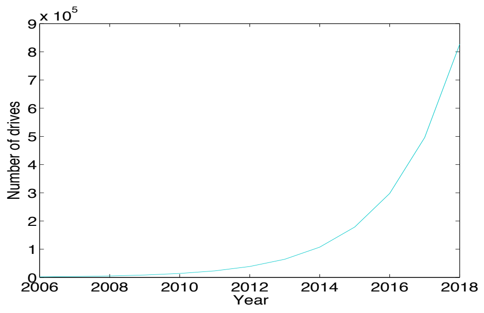

Figure 8. Number of drives in future systems.

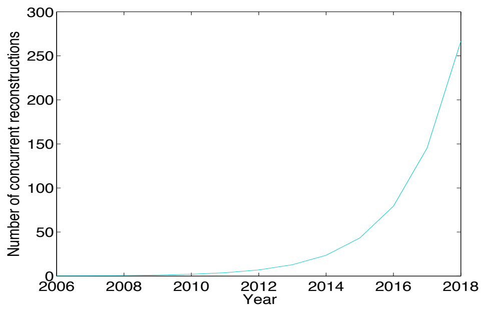

Figure 9. The number of concurrent reconstructions in the system.

5.8. Recommendations

The most demanding applications, often the same applications that justify the largest computers, will see increasing failure rates if the trends of top500.org continue. Using the standard checkpoint restart fault tolerance strategy, the effective utilization of petascale machines running demanding applications will fall off. Relying on machine manufacturing to counter this trend is not recommended by historical data, and relying on 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.

6. Petascale storage projections

Even if storage bandwidth in petascale computers is only required to maintain a balanced system design [14], difficult scaling issues for storage system designers are coming.

First, a individual disk bandwidth grows at a rate of about 20% per year, which is significantly slower than the 100% per year growth rate that top500.org predicts. To keep up, the number of disk drives in a system will have to increase at an impressive rate. Figure 8 projects the number of drives in a system necessary to (just) maintain balance. The figure shows that, if current technology trends continue, by 2018 an HPC system at the top of top500.org chart will need to have more than 800,000 disk drives. Managing this number of independent disk drives, much less delivering all of their bandwidth to an application, will be extremely challenging for storage system designers.

Second, disk drive capacity will keep growing by about 50% per year, thereby continuously increasing the amount of work needed to reconstruct a failed drive, and the time needed to complete this reconstruction. While other trends, such as decrease in physical size (diameter) of drives, will help to limit the increase in reconstruction time, these are single step decreases limited by the poorer cost effectiveness of the smaller disks. Overall we think it is realistic to expect an increase in reconstruction time of at least 10% per year. Assuming that today reconstruction times are often about 30 hours [2] and that 3% of drives in a system fail per year on average (recall Section 4.1), we can project the number of concurrent reconstructions going on in future HPC systems, as shown in Figure 9. The figure indicates that in the year 2018 on average nearly 300 concurrent reconstructions will be in progress at any time!

This analysis does not consider the current trend toward using more expensive error correcting codes in disk arrays in order to tolerate second failures during long reconstructions. It seems quite clear that petascale storage systems designers will be spending a large fraction of their efforts on fault tolerance inside the storage systems on which petascale application fault tolerance depends.

[2] The minimum reconstruction time would be shorter, on the order of 12 hours, but reconstruction is usually not performed at maximum speed in order to not interfere with foreground workload. For example, the most common disk array in use in HPC systems takes 8 days to do a reconstruction.

7. The computer failure data repository

The work described in this paper is part of our broader research agenda with the goal of analyzing and making publicly available failure data from a large variety of real production systems. We have built a public *Computer Failure Data Repository (CFDR)*, hosted by the USENIX association [1]. The goal of the repository is to accelerate research on system reliability by filling the nearly empty collection of public data with detailed failure data from a variety of large production systems. At this point, most of the organizations who have contributed the data described in Section 2 of this paper have agreed to make the data publicly available as part of the repository. Los Alamos National Laboratory and the National Energy Research Scientific Computing Center (NERSC) have already made their data available on their public web sites [2, 3]. We encourage all other petascale computing organizations to collect and publish failure data for their systems in the repository.

8. Related work

Large-scale studies of failures in real production systems are scarce, probably as a result of the reluctance of the owners of such systems to release failure data. Most existing studies are limited to only a few months of data, covering typically only a few hundred failures [16, 22, 23, 29, 34, 37]. Many of the most commonly cited studies on failure analysis stem from the late 80's and early 90's, when computer systems where significantly different from today [12, 13, 15, 18, 20, 21, 34]. More recently, several studies have been published on storage failures [32, 36, 25], all reporting ARR rates similar to the 3–6% range we observe in our data.

9. Acknowledgments

We would like to thank Jamez Nunez and Gary Grider from the High Performance Computing Division at Los Alamos National Lab and Katie Vargo, J. Ray Scott and Robin Flaus from the Pittsburgh Supercomputing Center for collecting and providing us with data and helping us to interpret the data. We also thank the National Energy Research Scientific Computing Center (NERSC) for making failure data collected on their systems publicly available. We also thank the other people and organizations, who have provided us with data, but would like to remain unnamed. For discussions relating to the use of high end systems, we would like to thank Mark Seager and Dave Fox of the Lawrence Livermore National Lab. We thank the members and companies of the PDL Consortium (including APC, Cisco, EMC, Hewlett-Packard, Hitachi, IBM, Intel, 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 [4] and on research sponsored in part by the Army Research Office, under agreement number DAAD19–02–1–0389.

10. Bibliography

[1] The Computer Failure Data Repository (CFDR). http://cfdr.usenix.org.
[2] The LANL raw data and more information is available at:. http://www.lanl.gov/projects/computerscience/data/.
[3] NERSC has made raw data and information on I/O related failures available at the following URL:. http://pdsi.nersc.gov/.
[4] Scientific Discovery through Advanced Computing (SciDAC), The Petascale Data Storage Institute (PDSI). http://www.pdsi-scidac.org, 2006.
[5] Top 500 supercomputing sites. http://www.top500.org, 2007.
[6] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.
[7] T. C. Bressoud and F. B. Schneider. Hypervisor-based fault tolerance. *ACM Trans. Comput. Syst.*, 14(1):80–107, 1996.
[8] J. Chapin, M. Rosenblum, S. Devine, T. Lahiri, D. Teodosiu, and A. Gupta. Hive: fault containment for shared-memory multiprocessors. In *SOSP '95: Proceedings of the fifteenth ACM symposium on Operating systems principles*, 1995.
[9] J. G. Elerath. AFR: problems of definition, calculation and measurement in a commercial environment. In *Proc. of the Annual Reliability and Maintainability Symposium*, 2000.
[10] J. G. Elerath. Specifying reliability in the disk drive industry: No more MTBFs. In *Proc. of the Annual Reliability and Maintainability Symposium*, 2000.

[11] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Comput. Surv.*, 34(3):375–408, 2002.

[12] J. Gray. Why do computers stop and what can be done about it. In *Proc. of the 5th Symposium on Reliability in Distributed Software and Database Systems*, 1986.

[13] J. Gray. A census of tandem system availability between 1985 and 1990. *IEEE Transactions on Reliability*, 39(4), 1990.

[14] G. Grider. 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, 2006.

[15] R. K. Iyer, D. J. Rossetti, and M. C. Hsueh. Measurement and modeling of computer reliability as affected by system activity. *ACM Trans. Comput. Syst.*, 4(3), 1986.

[16] M. Kalyanakrishnam, Z. Kalbarczyk, and R. Iyer. Failure data analysis of a LAN of Windows NT based computers. In *Proc. of the 18th IEEE Symposium on Reliable Distributed Systems*, 1999.

[17] K. Koch. The new roadrunner supercomputer: What, when, and how. In *Presentation at SC'06*.

[18] T.-T. Y. Lin and D. P. Siewiorek. Error log analysis: Statistical modeling and heuristic trend analysis. *IEEE Transactions on Reliability*, 39(4), 1990.

[19] D. McEvoy. 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.

[20] J. Meyer and L. Wei. Analysis of workload influence on dependability. In *Proc. International Symposium on Fault-Tolerant Computing*, 1988.

[21] B. Murphy and T. Gent. Measuring system and software reliability using an automated data collection process. *Quality and Reliability Engineering International*, 11(5), 1995.

[22] D. Nurmi, J. Brevik, and R. Wolski. Modeling machine availability in enterprise and wide-area distributed computing environments. In *Euro-Par'05*, 2005.

[23] D. L. Oppenheimer, A. Ganapathi, and D. A. Patterson. Why do internet services fail, and what can be done about it? In *USENIX Symposium on Internet Technologies and Systems*, 2003.

[24] D. Patterson, G. Gibson, and R. Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proc. of the ACM SIGMOD International Conference on Management of Data*, 1988.

[25] E. Pinheiro, W. D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *Proc. of the FAST '07 Conference on File and Storage Technologies*, 2007.

[26] J. S. Plank and K. Li. Faster checkpointing with N + 1 parity. In *Proc. 24th International Symposium on Fault Tolerant Computing*, 1994.

[27] J. S. Plank, K. Li, and M. A. Puening. Diskless checkpointing. *IEEE Trans. Parallel Distrib. Syst.*, 9(10):972–986, 1998.

[28] P. C. Roth. The Path to Petascale at Oak Ridge National Laboratory. In *Petascale Data Storage Workshop Supercomputing '06*, 2006.

[29] R. K. Sahoo, R. K., A. Sivasubramaniam, M. S. Squillante, and Y. Zhang. Failure data analysis of a large-scale heterogeneous server environment. In *Proc. of the 2004 International Conference on Dependable Systems and Networks (DSN'04)*, 2004.

[30] B. Schroeder and G. Gibson. 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.

[31] B. Schroeder and G. A. Gibson. Disk failures in the real world: what does an mttf of 1,000,000 hours mean to you? In *FAST'07: Proceedings of the 5th conference on USENIX Conference on File and Storage Technologies*, 2007.

[32] T. Schwarz, M. Baker, S. Bassi, B. Baumgart, W. Flagg, C. van Ingen, K. Joste, M. Manasse, and M. Shah. Disk failure investigations at the internet archive. In *Work-in-Progess session, NASA/IEEE Conference on Mass Storage Systems and Technologies (MSST2006)*, 2006.

[33] H. Simon. Petascale computing in the U.S.. Slides from presentation at the ACTS workshop. http://acts.nersc.gov/events/Workshop2006/slides/Simon.pdf, June 2006.

[34] D. Tang, R. K. Iyer, and S. S. Subramani. Failure analysis and modelling of a VAX cluster system. In *Proc. International Symposium on Fault-tolerant computing*, 1990.

[35] N. H. Vaidya. A case for two-level distributed recovery schemes. In *Proceedings of the 1995 ACM SIGMETRICS conference*, 1995.

[36] C. van Ingen and J. Gray. Empirical measurements of disk failure rates and error rates. In *MSR-TR-2005-166*, 2005.

[37] J. Xu, Z. Kalbarczyk, and R. K. Iyer. Networked Windows NT system field failure data analysis. In *Proc. of the 1999 Pacific Rim International Symposium on Dependable Computing*, 1999.

[38] J. Yang and F.-B. Sun. A comprehensive review of hard-disk drive reliability. In *Proc. of the Annual Reliability and Maintainability Symposium*, 1999.

[39] J. W. Young. A first order approximation to the optimum checkpoint interval. *Commun. ACM*, 17(9):530–531, 1974.