A Large-Scale Study of Failures in High-Performance Computing Systems

Bianca Schroeder and Garth A. Gibson, *IEEE*

Abstract—Designing highly dependable systems requires a good understanding of failure characteristics. Unfortunately, little raw data on failures in large IT installations are publicly available. This paper analyzes failure data collected at two large high-performance computing sites. The first data set has been collected over the past nine years at Los Alamos National Laboratory (LANL) and has recently been made publicly available. It covers 23,000 failures recorded on more than 20 different systems at LANL, mostly large clusters of SMP and NUMA nodes. The second data set has been collected over the period of one year on one large supercomputing system comprising 20 nodes and more than 10,000 processors. We study the statistics of the data, including the root cause of failures, the mean time between failures, and the mean time to repair. We find, for example, that average failure rates differ wildly across systems, ranging from 20-1000 failures per year, and that time between failures is modeled well by a Weibull distribution with decreasing hazard rate. From one system to another, mean repair time varies from less than an hour to more than a day, and repair times are well modeled by a lognormal distribution.

Index Terms—Large-scale systems, high-performance computing, supercomputing, reliability, failures, node outages, field study, empirical study, repair time, time between failures, root cause.

1 Introduction

Research in the area of dependable computing relies in many ways on a thorough understanding of what failures in real systems look like. For example, knowledge of failure characteristics can be used in resource allocation to improve cluster availability [5], [28]. The design and analysis of checkpoint strategies relies on certain statistical properties of failures [8], [24], [26]. Creating realistic benchmarks and testbeds for reliability testing requires an understanding of the characteristics of real failures.

Unfortunately, obtaining access to failure data from modern, large-scale systems is difficult, since such data is often perceived to be sensitive. Existing studies of failures are often based on only a few months of data, covering typically only a few hundred failures [22], [27], [16], [19], [15], [7]. Many of the commonly cited studies on failure analysis stem from the late 1980s and early 1990s, when computer systems were significantly different from today [3], [4], [6], [13], [22], [9], [11]. Finally, none of the raw data used in the above studies has been made publicly available for use by other researchers.

This paper analyzes failure data collected at two large high-performance computing sites. The first data set was collected over the past nine years at Los Alamos National Laboratory (LANL) and covers 22 high-performance computing (HPC) systems, including a total of 4,750 machines and 24,101 processors. The data contains an entry for any failure that occurred during the nine-year time period and that required the attention of a system administrator. For each failure, the data includes start time and end time, the system and node affected, as well as categorized root cause information. The second data set was collected at another supercomputing site, which prefers to remain anonymous, and covers one year of node outages at one large HPC system containing more than 10,000 processors. To the best of our knowledge, this is the largest set of failure data studied in the literature to date, both in terms of the time period spanned, and the number of systems and processors covered. Moreover, all LANL data used in this work have been made publicly available [1], presenting the first public release of a large set of failure data from large-scale production systems.

Our goal is to provide a description of the statistical properties of the data, as well as information for other researchers on how to interpret the publicly released data. We first describe the environments the data come from, including the systems and the workloads, the data collection process, and the structure of the data records (Section 2). Section 3 describes the methodology of our data analysis. We then study the data with respect to three important properties of system failures: the root causes (Section 4), the time between failures (Section 5), and the time to repair (Section 6). Section 7 compares our results to related work. Section 8 describes our ongoing efforts to create a public failure data repository and some of our experiences in obtaining failure data. Section 9 concludes.

- *B. Schroeder is with the Department of Computer Science, University of Toronto, Toronto, ON M5S 2E4, Canada. E-mail: bianca@cs.toronto.edu.*
- *G.A. Gibson is with the Computer Science Department, Carnegie Mellon University, Wean Hall 8219, 5000 Forbes Ave., Pittsburgh, PA 15213. E-mail: garth@cs.cmu.edu.*

Manuscript received 19 Mar. 2007; revised 7 Jan. 2008; accepted 7 Aug. 2008; published online 28 Jan. 2009.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2007-03-0040. Digital Object Identifier no. 10.1109/TDSC.2009.4.

2 Description of the Data and Environment

2.1 The Systems

We obtained data from two different high-performance computing sites, LANL and a supercomputing site that would like to remain unnamed.

TABLE 1
Overview of LANL Systems. Systems 1-18 are SMP-based, and systems 19-22 are NUMA-based

(I) High-level system information				(II) Information per node category			
HW	ID	Nodes	Procs	Procs /node	Production Time	Mem (GB)	NICs
A	1	1	8	8	N/A – 12/99	16	0
B	2	1	32	32	N/A – 12/03	8	1
C	3	1	4	4	N/A – 04/03	1	0
D	4	164	328	2	04/01 – now	1	1
				2	12/02 – now	1	1
E	5	256	1,024	4	12/01 – now	16	2
	6	128	512	4	09/01 – 01/02	16	2
	7	1,024	4,096	4	05/02 – now	8	2
				4	05/02 – now	16	2
				4	05/02 – now	32	2
				4	05/02 – now	352	2
	8	1,024	4,096	4	10/02 – now	8	2
				4	10/02 – now	16	2
				4	10/02 – now	32	2
	9	128	512	4	09/03 – now	4	1
	10	128	512	4	09/03 – now	4	1
	11	128	512	4	09/03 – now	4	1
	12	32	128	4	09/03 – now	4	1
				4	09/03 – now	16	1
F	13	128	256	2	09/03 – now	4	1
	14	256	512	2	09/03 – now	4	1
	15	256	512	2	09/03 – now	4	1
	16	256	512	2	09/03 – now	4	1
	17	256	512	2	09/03 – now	4	1
	18	512	1,024	2	09/03 – now	4	1
				2	03/05 – 06/05	4	1
G	19	16	2,048	128	12/96 – 09/02	32	4
				128	12/96 – 09/02	64	4
	20	49	6,152	128	01/97 – now	128	12
				128	01/97 – 11/05	32	12
				80	06/05 – now	80	0
	21	5	544	128	10/98 – 12/04	128	4
				32	01/98 – 12/04	16	4
				128	11/02 – now	64	4
				128	11/05 – 12/04	32	4
H	22	1	256	256	11/04 – now	1,024	0

The LANL data span 22 high-performance computing systems that have been in production use at LANL between 1996 and November 2005. Most of these systems are large clusters of either Nonuniform-Memory-Access (NUMA) nodes, or two-way and four-way Symmetric-Multiprocessing (SMP) nodes. In total, the systems include 4,750 nodes and 24,101 processors. Table 1 gives an overview of the 22 systems.

The left half of Table 1 provides high-level information for each system, including the total number of nodes and processors in the system, and a system ID we use throughout to refer to a system. The data do not include vendor-specific hardware information. Instead, it uses capital letters (A-H) to denote a system's processor/memory chip model. We refer to a system's label as its *hardware type*.

As the table shows, the LANL site has hosted a diverse set of systems. Systems vary widely in size, with the number of nodes ranging from 1 to 1,024 and the number of processors ranging from 4 to 6,152. Systems also vary in their hardware architecture. There is a large number of NUMA- and SMP-based machines, and a total of eight different processor and memory models (types A-H).

The nodes in a system are not always identical. While all nodes in a system have the same hardware type, they might differ in the number of processors and network interfaces (NICs), the amount of main memory, and the time they were in production use. The right half of Table 1 categorizes the nodes in a system with respect to these properties. For example, the nodes of system 12 fall into two categories, differing only in the amount of memory per node (4 versus 16 GB).

TABLE 2
Overview of System X

(I) High-level system information			(II) Information per node category			
HW	Nodes	Procs	Procs /node	Production Time	Mem (GB)	NICs
X	20	10,240	512	10/05 – now	1,024	N/A

The data from the unnamed site cover one year of node outages at a large supercomputing system, which we will refer to as system X. Table 2 provides an overview over the basic properties of system X. Note that the main difference compared to the LANL systems is that system X employs a very large number of processors per node (512) and that it was put in production relatively recently (October 2005).

2.2 The Workloads

At both sites, most workloads are large-scale long-running 3D scientific simulations, e.g., for nuclear stockpile stewardship or for plasma flow analysis. These applications perform long periods (often months) of CPU computation, interrupted every few hours by a few minutes of I/O for checkpointing. Simulation workloads are often accompanied by scientific visualization of large-scale data. Visualization workloads are also CPU-intensive, but involve more reading from storage than compute workloads. Finally, some nodes are used purely as front-end nodes, and others run more than one type of workload, e.g., graphics nodes often run compute workloads as well.

Failure tolerance is frequently implemented through periodic checkpointing. When a node fails, the job(s) running on it is stopped and restarted on a different set of nodes, either starting from the most recent checkpoint or from scratch if no checkpoint exists.

2.3 Data Collection

The LANL data are based on a "remedy" database created at LANL in June 1996. At that time, LANL introduced a site-wide policy that requires system administrators to enter a description of every failure they take care of into the remedy database. Consequentially, the database contains a record for every failure that occurred in LANL's HPC systems since June 1996 and that required intervention of a system administrator.

A failure record contains the time when the failure started, the time when it was resolved, the system and node affected, the type of workload running on the node, and the root cause. The workload is either *compute* for computational workloads, *graphics* for visualization workloads, or *fe* for front-end. Root causes fall in one of the following five high-level categories: *Human* error; *Environment*, including power outages or A/C failures; *Network* failure; *Software* failure; and *Hardware* failure. In addition, more detailed information on the root cause is captured, such as the particular hardware component affected by a *Hardware* failure. More information on the root causes can be found in

the released data [1]. The failure classification and rules for assigning failures to categories were developed jointly by hardware engineers, administrators, and operations staff.

Failure reporting at LANL follows the following protocol. Failures are detected by an automated monitoring system that pages operations staff whenever a node is down. The operations staff then create a failure record in the database specifying the start time of the failure, and the system and node affected, then turn the node over to a system administrator for repair. Upon repair, the system administrator notifies the operations staff who then put the node back into the job mix and fill in the end time of the failure record. If the system administrator was able to identify the root cause of the problem, he provides operations staff with the appropriate information for the "root cause" field of the failure record. Otherwise, the root cause is specified as "Unknown." Operations staff and system administrators have occasional follow-up meetings for failures with "Unknown" root cause. If the root cause becomes clear later on, the corresponding failure record is amended.

Two implications follow from the way the data were collected. First, these data are very different from the error logs used in many other studies. Error logs are automatically generated and track any exceptional events in the system, not only errors resulting in system failure. Moreover, error logs often contain multiple entries for the same error event.

Second, since the data were created manually by system administrators, the data quality depends on the accuracy of the administrators' reporting. Two potential problems in human-created failure data are underreporting of failure events and misreporting of root cause. For the LANL data, we do not consider underreporting (i.e., a failure does not get reported at all) a serious concern, since failure detection is initiated by automatic monitoring and failure reporting involves several people from different administrative domains (operations staff and system administrators). While misdiagnosis can never be ruled out completely, its frequency depends on the administrators' skills. LANL employs highly-trained staff backed by a well-funded cutting edge technology integration team, often pulling new technology into existence in collaboration with vendors; diagnosis can be expected to be as good as any customer and often as good as a vendor.

Failure data collection at the site containing system X works in a fashion similar to that described above for LANL. However, the data we obtained provided only monthly summary statistics, rather than detailed information on each individual outage. That is, for every month, our data set contains the number of failures observed during that month, including a breakdown of the number of failures by root cause. While the LANL data use six root cause categories (Hardware, Software, Network, Human, Environment, and Unknown), at system X failures are grouped into one of only four different categories: Hardware, Software, Human, and Unknown. Network failures in system X are assigned to either the Hardware, the Software, or the Human category, depending on which was responsible for the network failure. Node outages due to environmental problems were not observed in system X during the measurement period. For system X, no data on repair times is available.

3 Methodology

In our work, we use common statistical methods that would be covered in most college-level statistics classes [18]. We characterize an empirical distribution using three import metrics: the mean, the median, and the squared coefficient of variation (C^2). The squared coefficient of variation is a measure of variability and is defined as the squared standard deviation divided by the squared mean. The advantage of using the C^2 as a measure of variability, rather than the variance or the standard deviation, is that it is normalized by the mean, and hence allows comparison of variability across distributions with different means.

We also consider the empirical cumulative distribution function (CDF) and how well it is fit by four probability distributions commonly used in reliability theory[1]: the exponential, the Weibull, the gamma, and the lognormal distribution. We use maximum likelihood estimation to parameterize the distributions and evaluate the goodness of fit by visual inspection and the negative log-likelihood test.

Note that the goodness of fit that a distribution achieves depends on the degrees of freedom that the distribution offers. For example, a phase-type distribution with a high number of phases would likely give a better fit than any of the above standard distributions, which are limited to one or two parameters. Whenever the quality of fit allows, we prefer the simplest standard distributions, since these are well understood and simple to use. In our study, we have not found any reason to depend on more degrees of freedom.

4 Root Cause Breakdown

An obvious question when studying failures in computer systems is what caused the failures. Below, we study the entries in the high-level root cause field of the data.

We first look at the relative frequency of the high-level root cause categories in the LANL data and in system X, presented in Fig. 1. Recall that the LANL data provide a root cause breakdown of failures into human, environment, network, software, hardware, and unknown, while the failures in system X are attributed to either hardware, software, human, or unknown. Fig. 1 shows that the trends at both sites are similar. Hardware is the single largest component, with more than 50 percent of all failures assigned to this category for both the LANL systems and for system X. Software is the second largest contributor, with around 20 percent of all failures at both sites attributed to software.

It is important to note that at both sites, the number of failures with unidentified root cause is significant (14 percent and 23 percent, respectively). Since at both sites 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 insignificant.

While Fig. 1a provides an aggregate view of the root cause information across all LANL systems, two interesting

1. We also considered the Pareto distribution [25], [15], but did not find it to be a better fit than any of the four standard distributions.

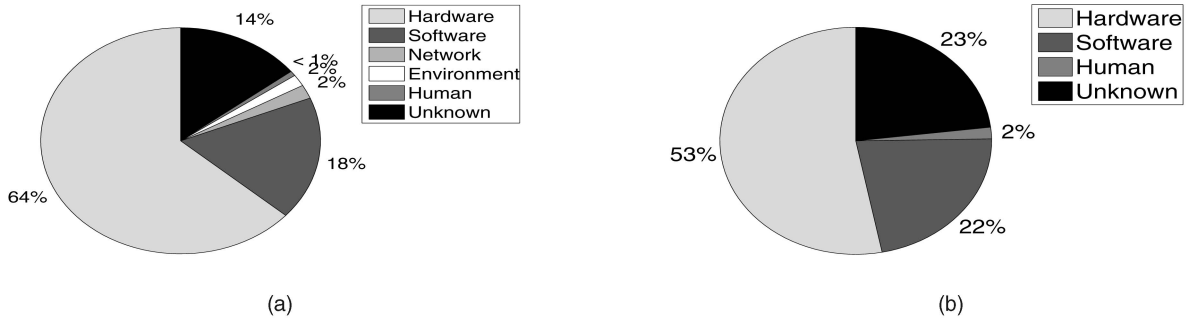


Fig. 1. The breakdown of failures into root causes for the LANL systems (a) and system X (b). The LANL graph shows the breakdown across all systems (A-H).

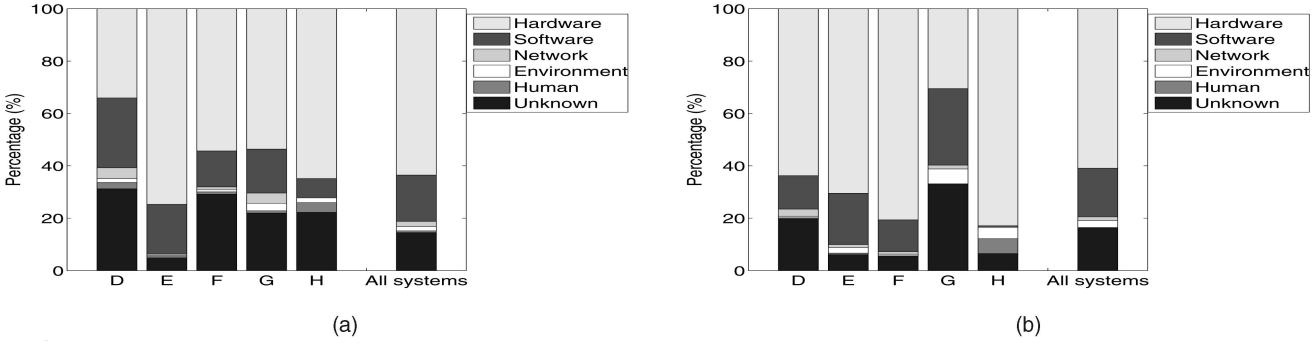


Fig. 2. The breakdown of failures into root causes (a) and the breakdown of downtime into root causes (b) for the LANL systems. Each graph shows the breakdown for systems of type D, E, F, G, and H and aggregate statistics across all systems (A-H).

questions are 1) whether the root cause breakdown changes over the lifetime of a system and 2) how it varies across the different types of systems at LANL.

Regarding question 1, we do not see a significant change in the root cause breakdown over the lifetime of a system (i.e., when moving from initial deployment to later years of operation). The main change we observe is that for some systems, the percentage of failures with unidentified root cause drops over time. However, the relative frequency of the other types of failures remains unchanged.

Regarding question (2), Fig. 2a shows root cause information broken down by the type of system. Each of the five bars to the left presents the breakdown across all failure records for systems of a particular hardware type.[2] The rightmost bar describes the breakdown across all failure records in the data set.

Fig. 2a indicates that while the basic trends are similar across system types, the actual breakdown varies. Hardware is the single largest component for each system type, with the actual percentage ranging from 30 percent to more than 60 percent. Software is the second largest contributor for each system type, with percentages ranging from five percent to 24 percent. Type D systems differ most from the other systems, in that hardware and software are almost equally frequent as a root cause.

We asked LANL about the higher fraction of downtime with unknown root cause for systems of type D and G and were told the reason lies in the circumstances surrounding their initial deployment. Systems of type G were the first NUMA-based clusters at LANL and were commissioned when LANL had just started to systematically record failure data. As a result, initially the fraction of failures with unknown root causes was high (>90 percent), but dropped to less than 10 percent within two years, as administrators gained more experience with the system and the root cause analysis. Similarly, the system of type D was the first large-scale SMP cluster at LANL, so initially the number of unknown root causes was high, but then quickly dropped.

The above example shows that interpreting failure data often requires interaction with the people who run the systems and collect the data. The public release of the data [1] includes a complete FAQ of all questions that we asked LANL in the process of our work.

We also study, using the repair time information in the LANL data, how much each root cause contributes to the total repair time. Fig. 2b shows the total downtime per system broken down by the downtime root cause. The basic trends are similar to the breakdown by frequency: hardware tends to be the single largest component, followed by software. Interestingly, for most systems, the failures with unknown root cause account for less than 5 percent of the total downtime despite the fact that the percentage of unknown root causes is higher. Only systems of type D and G have more than 5 percent of downtime with unknown root cause.

In addition to the six high-level root cause categories in the LANL data (recall Fig. 2), we also looked at the more detailed root cause information. Table 3 shows the 10 most

2. For better readability, we omit bars for types A-C, which are small single-node systems.

TABLE 3
Detailed Root Cause Breakdown of LANL Data

Hardware root causes (%)		Hardware root causes (%) without type E systems		Software root causes (%)		Environmental root causes (%)	
CPU	42.8	Memory Dimm	30.1	Other Software	30.0	Power Outage	48.4
Memory Dimm	21.4	Node Board	16.4	OS	26.0	UPS	21.2
Node Board	6.8	Other	11.8	Parallel File System	11.8	Power Spike	15.1
Other	5.1	Power Supply	9.7	Kernel software	6.0	Chillers	9.8
Power Supply	4.4	Interconnect Interface	6.6	Scheduler Software	4.9	Environment	5.3
Interconnect Interface	3.1	Interconnect Soft Error	3.1	Cluster File System	3.6		
Disk Drive	2.0	CPU	2.4	Resource Mgmt System	3.2		
Interconnect Soft Error	1.3	Fan Assembly	1.8	Network	2.7		
System Board	0.9	Router Board	1.5	User code	2.4		
PCI Backplane	0.8	Fibre Raid Controller	1.4	NFS	1.6		

common root causes within the high-level hardware, software, and environmental categories and their percentages.

As the first column in the table indicates, a large fraction (more than 40 percent) of all hardware-related outages at LANL are attributed to CPU. The second most commonly blamed hardware component is memory DIMMS. However, a closer look at the data reveals that for most systems memory is the by far more common root cause than CPU. The only exception are systems of type E. System E experienced a very high percentage (more than 50 percent) of CPU-related failures, due to a design flaw in the type E CPU.

The second column in the table presents the breakdown of hardware failures into low-level root causes across all LANL systems, *except for type E systems*, indicating that nearly a third of all hardware-related node failures among those systems are due to memory. In fact, we find that in all systems memory-related failures make up a significant portion of *all failures*, not just hardware failures. Memory was the single most common "low-level" root cause for all systems (across all failures not only hardware failures), except for system E. For each system covered by the data (including type E systems), more than 10 percent of all failures (not only hardware failures) were due to memory. Interestingly, we observe similar trends for system X. While we do not have detailed low-level root cause information for all outages in system X, we do have records of a significant number of memory-related outages (making up more than 20 percent of all outages).

Surprised by the high percentage of node outages attributed to memory DIMMS, we talked to the administrators of the systems about the nature of these outages. We were told that most node outages attributed to memory are not fatal hardware failures that require the replacement of a DIMM. Instead, it is common that the number of bit flips exceeds what the error correcting code is able to correct, causing the system to crash. One might at first assume that this is a problem particular to LANL, which happens to be geographically located at a high altitude, and hence, might see a higher rate of cosmic rays. However, as mentioned above, we also see similarly high DIMM failure rates for system X, which is located at a low altitude.

The third column in Table 3 provides a breakdown of software-related failures into low-level root cause categories. Unfortunately, a significant fraction of software failures were not specified further, but rather assigned to the "Other Software" category. When looking at individual systems, we find that the detailed breakdown for software-related failures varies quite a bit across systems. For system F, the most common software failure was related to the parallel file system, for system H to the scheduler software, and for system E to the operating system. For system D and G, a large portion of the software failures were not specified further ("Other Software").

The fourth column in Table 3 provides a breakdown of failures with environmental root cause. Nearly half of the failures in this category are due to power outages. The second most common root cause are problems with a uninterruptible power supply (UPS) device, followed by power spikes.

For network failures and human errors, the data does not contain a more detailed breakdown, and hence, there are no corresponding columns in Table 3.

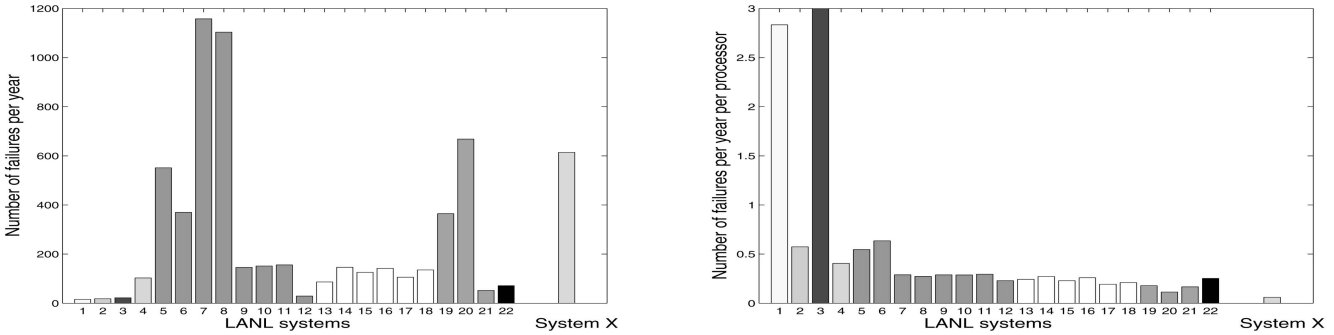
5 Analysis of Failure Rates

5.1 Failure Rate as a Function of System and Node

This section looks at how failure rates vary across different systems, and across the nodes within the same system. Studying failure rates across different systems is interesting since it provides insights on the effect of parameters such as system size and hardware type. Knowledge on how failure rates vary across the nodes in a system can be utilized in job scheduling, for instance, by assigning critical jobs or jobs with high recovery time to more reliable nodes.

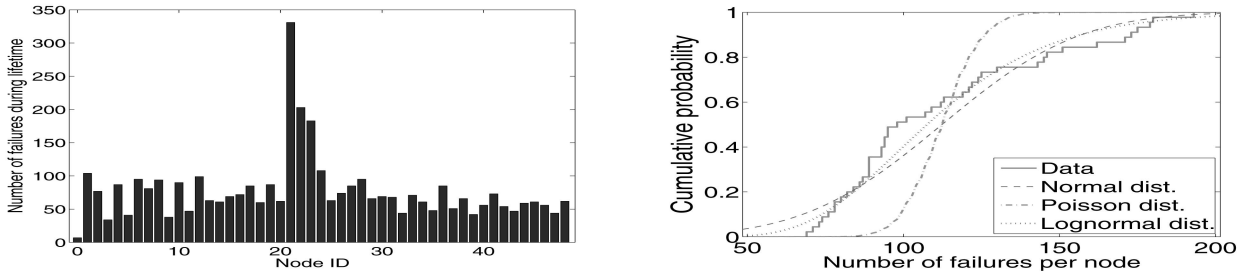
Fig. 3a shows for each of the 22 systems the average number of failures recorded per year during the system's production time. The yearly failure rate varies widely across systems, ranging from only 17 failures per year for system 2, to an average of 1,159 failures per year for system 7. In fact, variability in the failure rate is high even among systems of the same hardware type.

The main reason for the vast differences in failure rate across systems is that they vary widely in size. Fig. 3b shows for each system the average number of failures per year normalized by the number of processors in the system. The normalized failure rates show significantly less variability across systems, in particular, across systems with the



(a) (b)

Fig. 3. (a) Average number of failures for each 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.



(a) (b)

Fig. 4 (a) Number of failures per node for system 20 as a function of node ID. (b) The corresponding CDF, fitted with a Poisson, normal, and lognormal distribution.

same hardware type. For example, all type E systems (systems 5-12) exhibit a similar normalized failure rate,[3] although they range in size from 128 to 1,024 nodes. The same holds for type F systems (systems 13-18), which vary in size from 128 to 512 nodes. *This indicates that failure rates do not grow significantly faster than linearly with system size.*

We next concentrate on the distribution of failures across the nodes of a system. Fig. 4a shows the total number of failures for each node of system 20 during the entire system lifetime.[4] We first observe that nodes 21-23 experience a significantly higher number of failures than the other nodes. While nodes 21-23 make up only six percent of all nodes, they account for 20 percent of all failures. A possible explanation is that nodes 21-23 run different workloads than the other nodes in the system. Nodes 21-23 are the only nodes used for visualization, as well as computation, resulting in a more varied and interactive workload compared to the other nodes. We make similar observations for other systems, where failure rates vary significantly depending on a node's workload. For example, for systems E and F, the front-end nodes, which run a more varied, interactive workload, exhibit a much higher failure rate than the other nodes.

While it seems clear from Fig. 4a that the behavior of graphics nodes is very different from that of other nodes, another question is how similar the failure rates of the remaining (compute-only) nodes are to each other. Fig. 4b shows the CDF of the measured number of failures per node for compute-only nodes, with three different distributions fitted to it: the Poisson, the normal, and the lognormal distributions. If the failure rate at all nodes followed a Poisson process with the same mean (as often assumed, e.g., in work on checkpointing protocols), the distribution of failures across nodes would be expected to match a Poisson distribution. Instead, we find that the Poisson distribution is a poor fit, since the measured data have a higher variability than that of the Poisson fit. The normal and lognormal distribution are a much better fit, visually as well as measured by the negative log-likelihood. This indicates that the assumption of Poisson failure rates with equal means across nodes is suspect.

5.2 Failure Rate at Different Time Scales

Next, we look at how failure rates vary across different time scales, from very large (system lifetime) to very short (daily and weekly). Knowing how failure rates vary as a function of time is important for generating realistic failure workloads and for optimizing recovery mechanisms.

We begin with the largest possible time scale by looking at failure rates over the entire lifetime of a system. We find that for all systems in our data set, the failure rate as a function of system age follows one of two shapes. Fig. 5 shows a representative example for each shape.

The left graph in Fig. 5 shows the number of failures per month for system 5 at LANL, starting at production time. Failure rates are high initially, and then drop significantly during the first months. The shape of this curve is the most common one and is representative of all LANL systems of types E and F.

3. The higher failure rates for systems 5-6 are due to the fact that they were the first systems of type E at LANL and experienced a higher failure rate during the initial months of deployment.

4. Note that the lifetime of all nodes is the same, with the exception of node 0, which has been in production for a much shorter time (see Table 1).

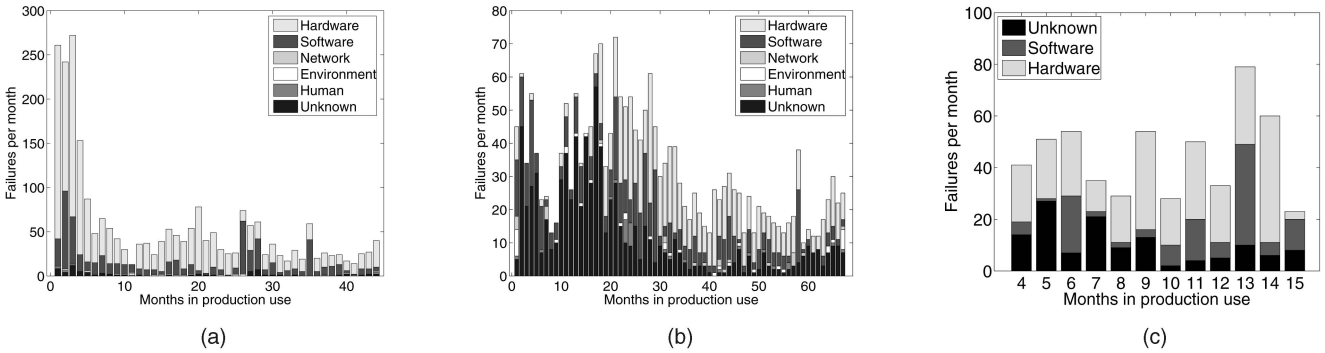


Fig. 5. Some representative examples for how the failure rate changes as a function of system age (in months). The leftmost graph corresponds to system 5, which is representative for LANL systems of types E and F. The graph in the middle corresponds to system 19, which is representative of LANL systems of types D and G. The rightmost graph corresponds to system X. (a) LANL system 5. (b) LANL system 19. (c) System X.

The shape of this curve is intuitive in that the failure rate drops during the early age of a system, as initial hardware and software bugs are detected and fixed and administrators gain experience in running the system. One might wonder why the initial problems were not solved during the typically 1-2 months of testing *before* production time. The reason most likely is that many problems in hardware, software, and configuration are only exposed by real user code in the production workloads.

The middle graph in Fig. 5b corresponds to the failures observed over the lifetime of system 19 and represents the other commonly observed shape. The shape of this curve is representative for systems of types D and G, and is less intuitive: The failure rate actually grows over a period of nearly 20 months, before it eventually starts dropping. One possible explanation for this behavior is that getting these systems into full production was a slow and painful process.

Type G systems were the first systems of the NUMA era at LANL and the first systems anywhere that arranged such a large number of NUMA machines in a cluster. As a result, the first two years involved a lot of development work among system administrators, vendors, and users. Administrators developed new software for managing the system and providing the infrastructure to run large parallel applications. Users developed new large-scale applications that would not have been feasible to run on previous systems. With the slower development process, it took longer until the systems were running the full variety of production workloads and the majority of the initial bugs were exposed and fixed. The case for the type D system was similar in that it was the first large-scale SMP cluster at the site.

Two other observations support the above explanation. First, the failure rate curve for other SMP clusters (systems of types E and F) that were introduced after type D and were running full production workloads earlier in their life, follows the more traditional pattern in Fig. 5a. Second, the curve of system 21, which was introduced two years after the other systems of type G, is much closer to Fig. 5a.

The rightmost graph in Fig. 5 corresponds to system X. Unlike the other two graphs in the figure, the shape of the graph for system X shows no signs of increased failure rates in the early lifetime of the system. However, for system X, we have no data available for the first three months of production use. The figure plots the failure rates only starting in month 4 of production use. It is likely that the failure rates of system X follow a pattern similar to the one observed in Fig. 5a, although at this point we are not able to validate this assumption.

Next, we look at how failure rates vary over smaller time scales. It is well known that usage patterns of systems vary with the time of the day and the day of the week. The question is whether there are similar patterns for failure rates. Fig. 6 categorizes all failures in the LANL data by hour of the day and by day of the week. We observe a strong correlation in both cases. During peak hours of the day, the failure rate is two times higher than at its lowest during the night. Similarly, the failure rate during weekdays is nearly two times as high as during the weekend. We interpret this as a correlation between a system's failure rate and its workload, since, in general, usage patterns (not specifically LANL), workload intensity, and the variety of workloads are lower during the night and on the weekend.

Another possible explanation for the observations in Fig. 6 would be that failure rates during the night and weekends are not lower, but that the detection of those failures is delayed until the beginning of the next (week-) day. We rule this explanation out, since failures are detected by an automated system, and not by users or administrators. Moreover, if delayed detection was the reason, one would expect a large peak on Mondays, and lower failure rates on the following days, which is not what we see.

5.3 Statistical Properties of Time Between Failures

In this section, we view the sequence of failure events as a stochastic process and study the distribution of its interarrival times, i.e., the time between failures. Since the data from system X provide only summary statistics, and no timestamps for individual failures, we will focus in this

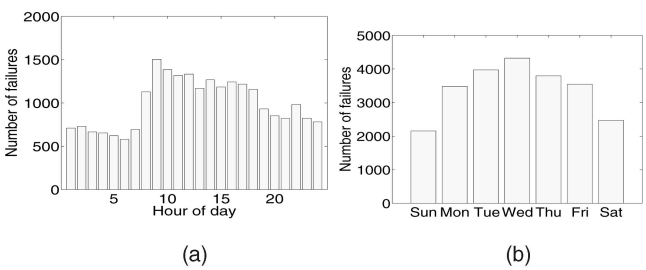


Fig. 6. Number of failures by hour of the day (a) and the day of the week (b) at LANL.

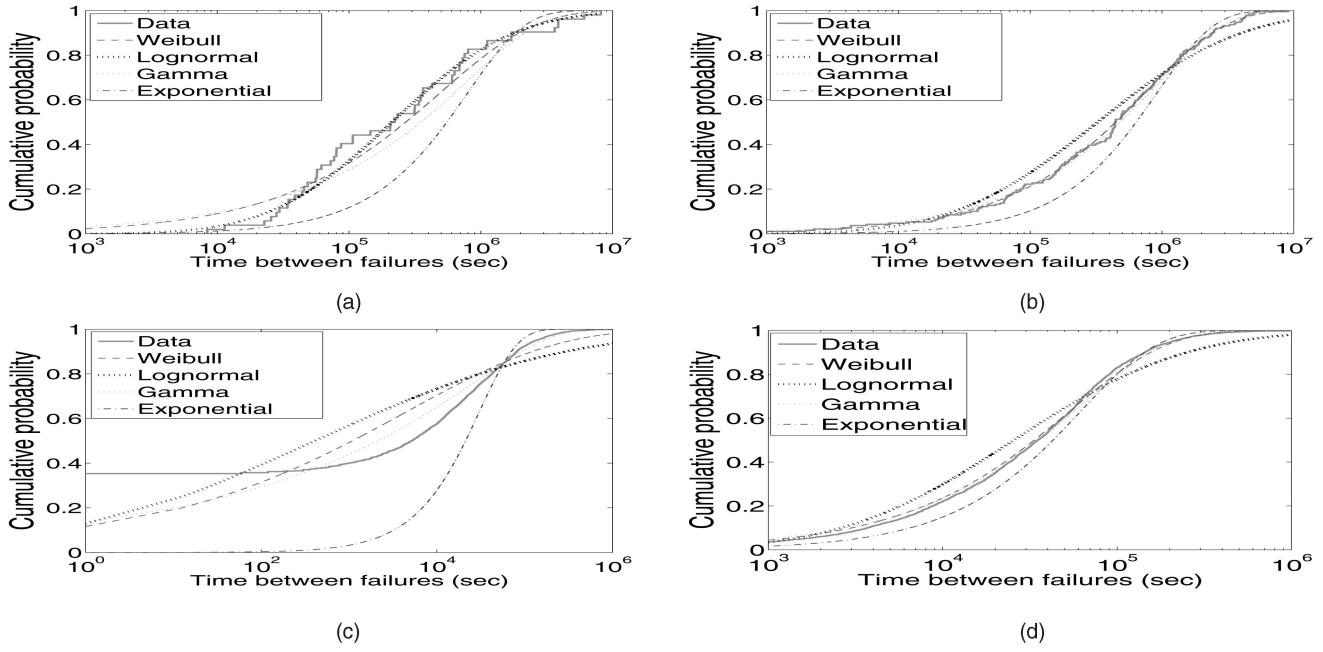


Fig. 7. Empirical CDF for interarrival times of failures on node 22 in system 20 at LANL early in production (a) and late in production (b) Empirical CDF for interarrival times of failures for the system-wide view of failures in system 20 at LANL early in production (c) and late in production (d).

section entirely on the LANL data. We take two different views of the failure process: 1) the view as seen by an individual node, i.e., we study the time between failures that affect only this particular node; 2) and the view as seen by the whole system, i.e., we study the time between subsequent failures that affect any node in the system.

Since failure rates vary over a system's lifetime (Fig. 5), the time between failures also varies. We therefore analyze the time between failures separately for the early production time, when failure rates are high, and the remaining system life, when failure rates are lower. Throughout, we focus on system 20 as an illustrative example.

We begin with the view of the time between failures as seen by an individual node. Figs. 7a and 7b show the empirical distribution at node 22 in system 20 during the years 1996-1999 and the years 2000-2005, respectively, fitted by four standard distributions. We see that from 2000-2005, the distribution between failures is well modeled by a Weibull or gamma distribution. Both distributions create an equally good visual fit and the same negative log-likelihood. The simpler exponential distribution is a poor fit, as its C^2 of 1 is significantly lower than the data's C^2 of 1.9.

During years 1996-1999, the empirical distribution of the time between failures at node 22 looks quite different (Fig. 7a) from the 2000-2005 period. During this time, the best fit is provided by the lognormal distribution, followed by the Weibull and the gamma distribution. The exponential distribution is an even poorer fit during the second half of the node's lifetime.

Next, we move to the system-wide view of the failures in system 20, shown in Figs. 7c and 7d. The basic trend for 2000-05 (Fig. 7d) is similar to the per node view during the same time. The Weibull and gamma distribution provide the best fit, while the lognormal and exponential fits are significantly worse.

The system-wide view during years 1996-1999 (Fig. 7c) exhibits a distribution that is very different from the others we have seen and is not well captured by any of the standard distributions. The reason is that an exceptionally large number (>30 percent) of interarrival times are zero, indicating a simultaneous failure of two or more nodes. While we did not perform a rigorous analysis of correlations between nodes, this high number of simultaneous failures indicates the existence of a tight correlation in the initial years of this cluster.

Given that in all the above cases the exponential distribution does not provide a good fit to the data, one might ask how the empirical distribution differs from an exponential distribution, i.e., what are the properties of the data that the exponential distribution cannot capture.

We identify as a first differentiating property that the time between failures in the data has a significantly higher variability than that of an exponential distribution. For example, the time between failures at node 22 in years 1996-1999 has a squared coefficient of variation C^2 of 3.9, while the C^2 of an exponential distribution is always 1. For some nodes and systems, we observe C^2 values as high as 35.

We identify as a second differentiating property that the empirical time between failures exhibits *decreasing hazard rates*. For failure interarrival distributions, the hazard rate measures how the time since the last failure influences the expected time until the next failure. An increasing hazard rate function predicts that if the time since a failure is long then the next failure is coming soon. And a decreasing hazard rate function predicts the reverse.

For an exponential distribution, the hazard rate function is constant, that is the probability of a failure does not depend on the time that has passed since the last failure. On the other hand, we find that in those cases in our previous discussion, where the Weibull distribution provides a good fit (Figs. 7a, 7b, and 7d), the shape parameter of the Weibull fit to the data is less than 1, which indicates that the hazard rate function is decreasing, i.e. not seeing a failure for a long time decreases the chance of seeing one in the near future.

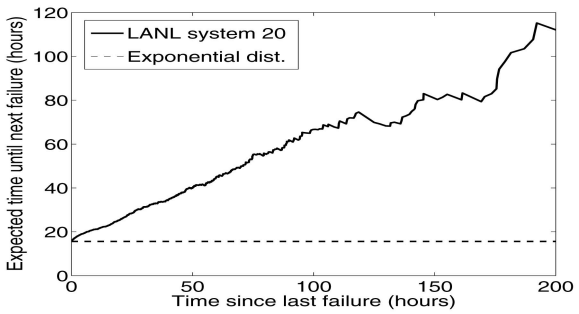


Fig. 8. Illustration of decreasing hazard rates in the LANL data versus constant hazard rates in the exponential distribution.

For example, the data in Fig. 7b are well fit by a Weibull distribution with shape parameter 0.7.

Fig. 8 provides a graphical illustration of the decreasing hazard rates in the time between failures. In the figure, we plot the expected time until the next failure as a function of the time since the last failure, for the system-wide view of time between failures for system 20 and for an exponential distribution fit to the data. For the exponential distribution, the time until the next failure is independent of the time since the last failure (dashed line). On the other hand, for the empirical data, the expected time until failure grows significantly with the time the system has survived without a failure (solid line).

5.4 Correlations Between Failures

We are interested in two different types of correlations, spatial and temporal. By spatial correlation, we mean correlations between failures of different nodes in the same system, i.e., does the fact that one node in a system fails increase the probability of observing another node failing in the same time interval. By temporal correlation, we mean correlations between the number of failures observed in a system in consecutive time intervals.

In order to test for spatial correlation, we compute six different time sequences for each node in a system, containing for each hour of operation[5] the number of observed failures due to hardware, software, environment, human error, network, and unknown root cause, respectively. We then compute the correlation coefficients between the different vectors. The only significant correlation (correlation coefficient larger than 0.1 and p-value smaller than 0.05) we observe is between failures that are due to network problems. There is no clear indication for correlations between other types of failures.

In order to study the degree of temporal correlation, we make use of the concept of autocorrelation. The autocorrelation function (ACF) measures the correlation of a random variable with itself at different time lags l. The ACF can, for example, be used to determine whether the number of failures in one day is correlated with the number of failures observed l days later. The autocorrelation coefficient can range between 1 (high positive correlation) and –1 (high negative correlation). For a stationary failure process (e.g., data coming from a Poisson process), the autocorrelation would be close to zero at all lags.

5. We also experimented with number of failures per day instead of per hour, with the same results.

Fig. 9 shows the autocorrelation function for the number of failures observed per day, per week, and per month, respectively, in system 19 broken down by the six different types of root cause. As the graphs show, significant autocorrelation exists at all three time granularities, indicating that the number of failures observed in one time interval is predictive of the number of failures expected in the following time intervals. Autocorrelation is strongest at the monthly granularity with autocorrelation coefficients higher than 0.8, but is still significant at weekly and monthly granularity, with autocorrelation coefficients higher than 0.2.

One might argue that the observed autocorrelation, in particular, at larger time granularities, is not surprising, given that we have shown earlier (recall Fig. 5) that failure rates change as a function of system age, in particular, during the time of initial deployment. In order to verify that the observed autocorrelation is not solely due to this factor, we repeated our analysis separately for the different parts of a system's lifetime. We find that the ACF graphs look nearly identical when produced for only the steady state part of a system's lifetime compared to the entire lifetime. Moreover, we find this to be the case for both systems that follow more closely the typical bathtub curve, such as system 5, as well as for other systems, such as system 19.

It is interesting to observe that autocorrelation varies greatly depending on the root cause of failures. Autocorrelation is strongest for failures that are due to hardware, software, and unknown root cause and is less significant for all other root causes (environment, network, human).

While Fig. 9 shows results only for system 19, we find that the trends are very similar for the other LANL systems, in terms of the strength of the observed autocorrelation. The only significant difference in the ACF of different systems is in the correlation behavior of failures with unknown root cause. For example, while we observe significant autocorrelation for failures with unknown root cause for system 19, failures with unknown root cause in system 5 show very little autocorrelation. One possible explanation might be that the actual root cause breakdown of failures with unknown root cause does vary across systems, e.g., in system 19, failures with unidentified root cause might be largely due to hardware failures (which exhibit high autocorrelation), while in system 5, they might be more likely due to network, human, or environmental problems (which exhibit low autocorrelation).

6 Analysis of Repair Times

A second important metric in system reliability is the time to repair. We first study how parameters such as the root cause of a failure and system parameters affect repair times. We then study the statistical properties of repair times, including their distribution and variability. Since the data on system X do not include information on repair times, we focus in this section entirely on the LANL data.

Table 4 shows the median and mean time to repair as a function of the root cause, and as an aggregate across all failure records in the LANL data. We find that both the median and the mean time to repair vary significantly depending on the root cause of the failure. The mean time to repair ranges from less than three hours for failures caused by human error, to nearly 10 hours for failures due to environmental problems. The mean time to repair for the

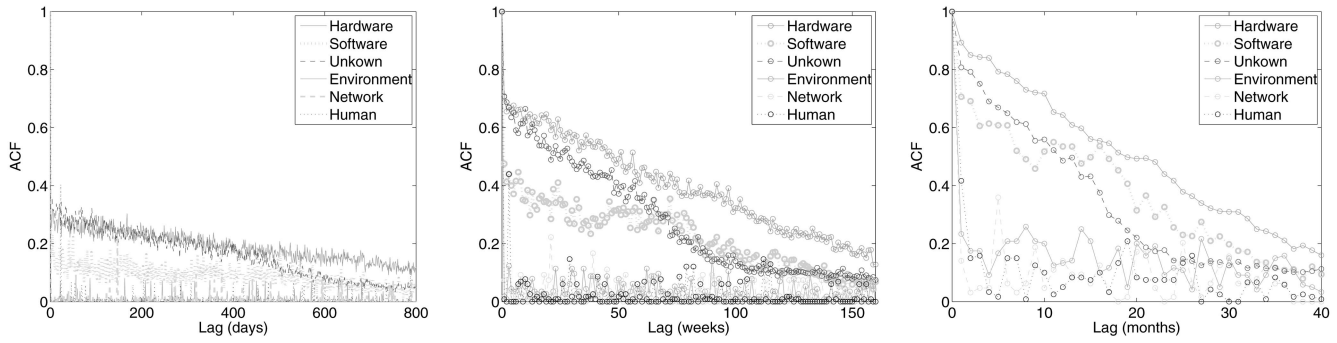


Fig. 9. The autocorrelation function for number of failures observed per day, per week, and per month, respectively.

other root cause categories varies between four and six hours. The mean repair time across all failures (independent of root cause) is close to six hours. The reason is that it is dominated by hardware and software failures which are the most frequent types of failures and exhibit mean repair times around six hours.

An important observation is that the time to repair for all types of failures is extremely variable, except for environmental problems. For example, in the case of software failures, the median time to repair is about 10 times lower than the mean, and in the case of hardware failures, it is four times lower than the mean. This high variability is also reflected in extremely high C^2 values (see bottom row of Table 4).

One reason for the high variability in repair times of software and hardware failures might be the diverse set of problems that can cause these failures. For example, the root cause information for hardware failures spans 99 different categories, compared to only two (power outage and A/C failure) for environmental problems. To test this hypothesis, we determined the C^2 for several types of hardware problems. We find that even within one type of hardware problem, variability can be high. For example, the C^2 for repair times of CPU, memory, and node interconnect problems is 36, 87, and 154, respectively. This indicates that there are other factors contributing to the high variability.

Fig. 12 shows the empirical CDF for all repair times in the data, and four standard distributions fitted to the data. The exponential distribution is a very poor fit, which is not surprising given the high variability in the repair times. The lognormal distribution is the best fit, both visually as well as measured by the negative log-likelihood. The Weibull distribution and the gamma distribution are weaker fits than the lognormal distribution, but still considerably better than the exponential distribution.

In Fig. 10, we consider how repair times vary across LANL systems. The graphs in Figs. 10a and 10b show the mean and median time to repair for each system, respectively. The figure indicates that the hardware type has a major effect on repair times. While systems of the same hardware type exhibit similar mean and median time to repair, repair times vary significantly across systems of different types.

Fig. 10 also indicates that system size is not a significant factor in repair time. For example, type E systems range from 128 to 1,024 nodes, but exhibit similar repair times. In fact, the largest type E systems (systems 7-8) are among the ones with the lowest median repair time.

The relatively consistent repair times across systems of the same hardware type are also reflected in the empirical CDF. We find that the CDF of repair times from systems of the same type (not shown in figure) is less variable than that across all systems, which results in an improved (albeit still suboptimal) exponential fit.

Finally, another interesting question is how repair times change over the lifetime of a system. Recall from Section 5 that failure rates can vary drastically over the span of a system's lifetime. We observe that repair times also change over time; however, the trends, are not as clear as for failure rates. We do observe, though, that repair times are consistently higher during the first year of operation compared to the remaining years. After the first one to two years of operations, the repair times are relatively stable.

Fig. 11 shows the mean and median repair time, broken down by root cause, for the first year of operation and the remaining years of operation. As the graphs indicate, mean repair time drops after the first year of operation for all types of failures, except for those with unknown root cause, often by more than a factor of two. Median repair times drop significantly for all types of failures after the first year of operation. The overall drop in mean repair time, across all failure types, is more than a factor of two, while the overall median repair time drops by more than a third. We also check whether variability in repair times changes over time, but do not see a significant change (results not shown in figure).

The decrease in repair times over time likely reflect the learning curve of the system administrators in troubleshooting problems in a new system. It might indicate that during the first year of operation, system administrators get significantly better at quickly identifying the root cause of a problem and fixing it.

TABLE 4
Statistical Properties of Time to Repair as a Function of the Root Cause of the Failure in the LANL Data

	Unkn.	Hum.	Env.	Netw.	SW	HW	All
Mean (min)	398	163	572	247	369	342	355
Median (min)	32	44	269	70	33	64	54
Std. Dev. (min)	6099	418	808	720	6316	4202	4854
Variability (C^2)	234	6	2	8	293	151	187

7 Comparison with Related Work

Work on characterizing failures in computer systems differs in the type of data used; the type and number of systems under study; the time of data collection; and the number of

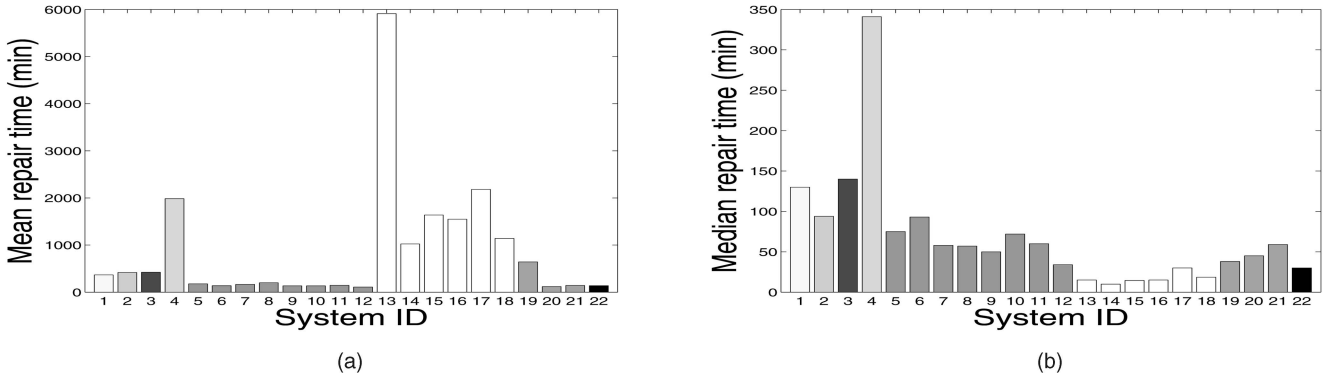


Fig. 10. Empirical CDF of repair times in the LANL data.

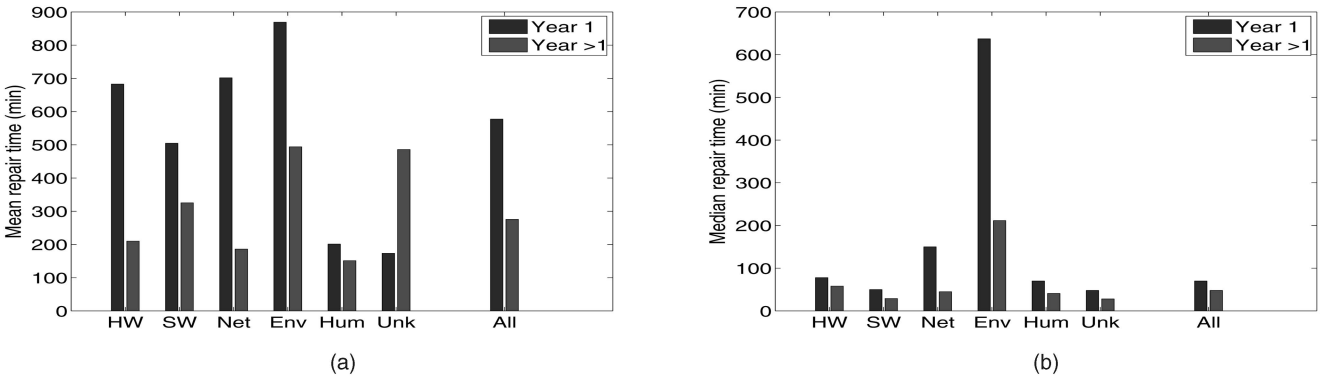


Fig. 11. Mean repair time (a) and median repair time (b) for each LANL system.

failure or error records in the data set. Table 5 gives an overview of several commonly cited studies of failure data.

Four of the above studies include root cause statistics [4], [13], [16], [7]. The percentage of software-related failures is reported to be around 20 percent [3], [13], [16] to 50 percent [4], [7]. Hardware is reported to make up 10-30 percent of all failures [4], [13], [16], [7]. Environment problems are reported to account for around 5 percent [4]. Network problems are reported to make up between 20 percent [16] and 40 percent [7]. Gray [4] reports 10-15 percent of problems due to human error, while Oppenheimer et al. [16] report 14-30 percent. The main difference to our results is the lower percentage of human error and network problems in our data. There are two possible explanations. First, the root cause of 20-30 percent of failures in our data is unknown and could lie in the human or network category. Second, the LANL environment is an expensive, very controlled environment with national safety obligations and priorities, so greater resources may be put into its infrastructure than is put into commercial environments.

Several studies analyze the time between failures [19], [22], [5], [27], [15], [10]. Four of the studies use distribution fitting and find the Weibull distribution to be a good fit [5], [27], [9], [15], which agrees with our results. Several studies also looked at the hazard rate function, but came to different conclusions. Some of them [5], [27], [9], [15] find decreasing hazard rates (Weibull shape parameter < 0.5). Others find that hazard rates are flat [22], or increasing [19]. We find decreasing hazard rates with Weibull shape parameter of 0.7-0.8.

Three studies [2], [6], [19] report correlations between workload and failure rate. Sahoo [19] reports a correlation between the type of workload and the failure rate, while Iyer [6] and Castillo [2] report a correlation between the workload intensity and the failure rate. We find evidence for both correlations, in that we observe different failure rates for compute, graphics, and front-end nodes, for different hours of the day and days of the week.

Sahoo et al. [19] also study the correlation of failure rate with hour of the day and the distribution of failures across nodes and find even stronger correlations than we do. They report that less than 4 percent of the nodes in a machine room experience almost 70 percent of the failures and find failure rates during the day to be four times higher than during the night.

Three studies [2], [6], [19] report correlations between workload and failure rate. Sahoo [19] reports a correlation between the type of workload and the failure rate, while Iyer [6] and Castillo [2] report a correlation between the

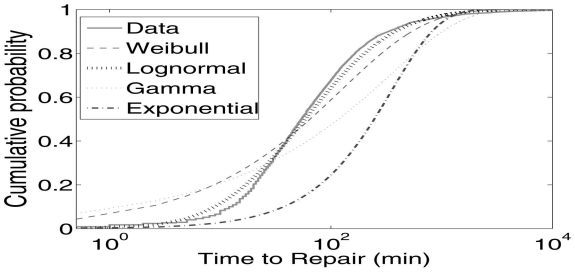


Fig. 12. Mean repair time (a) and median repair time (b) during the first year of operation versus the remaining years of a system's lifetime, broken down by root cause.

TABLE 5
Overview of Related Studies

Study	Date	Length	Environment	Type of Data	# Failures	Statistics
[3, 4]	1990	3 years	Tandem systems	Customer data	800	Root cause
[7]	1999	6 months	70 Windows NT mail server	Error logs	1100	Root cause
[16]	2003	3-6 months	3000 machines in Internet services	Error logs	501	Root cause
[13]	1995	7 years	VAX systems	Field data	N/A	Root cause
[22]	1990	8 months	7 VAX systems	Error logs	364	TBF
[9]	1990	22 months	13 VICE file servers	Error logs	300	TBF
[6]	1986	3 years	2 IBM 370/169 mainframes	Error logs	456	TBF
[19]	2004	1 year	395 nodes in machine room	Error logs	1285	TBF
[5]	2002	1-36 months	70 nodes in university and Internet services	Error logs	3200	TBF
[27]	1999	4 months	503 nodes in corporate envr.	Error logs	2127	TBF
[15]	2005	6–8 weeks	300 university cluster and Condor[23] nodes	Custom monitoring	N/A	TBF
[10]	1995	3 months	1170 internet hosts	RPC polling	N/A	TBF,TTR
[2]	1980	1 month	PDP-10 with KL10 processor	N/A	N/A	TBF,Utilization

workload intensity and the failure rate. We find evidence for both correlations, in that we observe different failure rates for compute, graphics, and front-end nodes, for different hours of the day and days of the week.

Sahoo et al. [19] also study the correlation of failure rate with hour of the day and the distribution of failures across nodes and find even stronger correlations than we do. They report that less than 4 percent of the nodes in a machine room experience almost 70 percent of the failures and find failure rates during the day to be four times higher than during the night.

We are not aware of any studies that report failure rates over the entire lifetime of large systems. However, there exist commonly used models for individual software or hardware components. The failures over the lifecycle of hardware components are often assumed to follow a "bathtub curve" with high failure rates at the beginning (infant mortality) and the end (wear out) of the lifecycle. The failure rate curve for software products is often assumed to drop over time (as more bugs are detected and removed), with the exception of some spikes caused by the release of new versions of the software [13], [12]. We find that the failure rate over the lifetime of large-scale HPC systems can differ significantly from the above two patterns (recall Fig. 5).

Repair times are studied only by Long et al. [10]. Long et al. estimate repair times of Internet hosts by repeated polling of those hosts. They, like us, conclude that repair times are not well modeled by an exponential distribution, but do not attempt to fit other distributions to the data.

In our recent work, we have collected and analyzed a number of data sets on storage failures, in particular, data on hard drive replacements in large storage systems [21]. We find that some of our findings on hard drive replacements parallel those we report in this paper on cluster node outages, while some characteristics of storage failures differ from those of cluster node failures. For example, the distribution of time between hard drive replacements is similar to the distribution of time between cluster node failures. The time between hard drive replacements is not exponentially distributed, but instead better modeled by a Weibull distribution. Again, the Weibull shape parameter for the best distribution fit to the data is in the range of 0.7-0.8. On the other hand, we find that failure rate as a function of system age behaves very differently for storage systems than for HPC cluster systems. Specifically, for storage systems, we find little evidence of infant mortality, but significant evidence of early wear out (increasing failure rates with system age).

An interesting question that is beyond the scope of our work is how system design choices depend on failure characteristics. Plank et al. [17] study how checkpointing strategies are affected by the distribution of time between failures, and Nath et al. [14] study how correlations between failures affect data placement in distributed storage systems.

An earlier version of the work presented in this paper has appeared in the International Conference on Dependable Systems and Networks (DSN '06) [20].

8 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 are currently working on creating a public *Computer Failure Data Repository (CFDR)*, to be hosted by the USENIX Association. 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. Below, we first briefly describe the data sets we have collected so far. We then describe some of our experiences from collecting failure data hoping that those might help others in obtaining data. Finally, we discuss our ongoing efforts and the long-term goals of the CFDR.

8.1 Data Sets

At this point, all LANL failure data that we have analyzed in this paper are publicly available. For some of the LANL systems, additional data have been made available, including event logs and usage data. We are currently working with the organization hosting system X to release their data as well. In addition to data on node outages from high-performance computing environments, we have also collected a number of failure data sets on storage failures, in particular, data on hard drive replacements in large storage systems [21].

Interestingly, we find that many of our findings on hard drive replacements parallel those we report in this paper on cluster node outages. For example, we find that the time between hard drive replacements is not exponentially distributed, but, instead, better modeled by a Weibull distribution. Again, the Weibull shape parameter for the

best distribution fit to the data is in the range of 0.7-0.8.

8.2 Experiences from Data Collection

Obtaining failure data is difficult, since these data are often perceived to be sensitive. In our pursuit to create a public failure data repository, we have talked to more than a dozen companies and high-performance computing labs about contributing data. Our experiences in this process have led us to believe that it is unlikely to obtain data from *vendors* of IT equipment, due to their fear of negative marketing. However, obtaining data from end users of IT equipment is often complicated by the perception that the data collection is not state-of-the-art and the complexity and efforts involved in collecting and explaining the data.

We found that sites with large amount of equipment are motivated to share data since they are facing a pressing need to provide reliability at scale and hope that researchers will be able to develop better solutions, if given real data to work with.

We would like to encourage any organizations that are collecting failure data for their systems to consider contributing to the repository. We would also like to encourage other researchers to share failure data sets that they have used in their work.

8.3 Long-Term Goals

Our work on the computer failure data repository focuses on three long-term goals. Our first goal is to extend the number of failure data sets hosted by the CFDR to cover a large, diverse set of sites. We are also pursuing other types of data, including usage data (job logs and utilization measurements) and event logs, to facilitate the study of correlations between such data and system failures. For the LANL systems, both usage data and event logs have recently been made publicly available.

Second, we plan to study the existing data sets in more detail, with a focus on how the results can be used for better or new techniques for avoiding, coping, and recovering from failures. For example, from the results in this work as well as in our related paper on hard drive failures, we find that several common assumptions about failure processes (e.g., i.i.d. exponentially distributed time between failures) are not realistic in practice. One path for future work is to reexamine algorithms and techniques for fault-tolerant systems to understand where unrealistic assumptions result in poor design choices and for those cases explore new algorithms.

Third, we hope that our experiences from working with a variety of sites on collecting and analyzing failure data will lead to some *best practices* for failure data collection. Currently, data collection and analysis are complicated by the fact that there is no widely accepted format for anomaly data and there exist no guidelines on what data to collect and how. Providing such guidelines will make it easier for sites to collect data that are useful and comparable across sites.

9 Summary

Many researchers have pointed out the importance of analyzing failure data and the need for a public failure data repository [16]. This paper provides a study of a large amount of failure data that have been collected at two large high-performance computing sites and have, in part, been made publicly available [1]. We hope that this data might serve as a first step toward a public data repository and encourage efforts at other sites to collect and clear data for public release. Below, we summarize a few of our findings.

- Failure rates vary widely across systems, ranging from 20 to more than 1,000 failures per year, and depend mostly on system size and less on the type of hardware.
- Failure rates are roughly proportional to the number of processors in a system, indicating that failure rates are not growing significantly faster than linearly with system size.
- There is evidence of a correlation between the failure rate of a machine and the type and intensity of the workload running on it. This is in agreement with earlier work for other types of systems [2], [6], [19].
- The curve of the failure rate over the lifetime of an HPC system looks often very different from lifecycle curves reported in the literature for individual hardware or software components.
- Time between failure is not modeled well by an exponential distribution, which agrees with earlier findings for other types of systems [5], [27], [9], [15], [19]. We find that the time between failure at individual nodes, as well as at an entire system, is fit well by a gamma or Weibull distribution with decreasing hazard rate (Weibull shape parameter of 0.7-0.8).
- Failures exhibit significant levels of temporal correlation at both short and long time lags. We find indication of autocorrelation for all types of failures; however, autocorrelation is particularly strong for hardware and software failures.
- We also find indication of spatial correlation, i.e., correlation between failures at different nodes during the same time interval. However, those are limited to failures with network root cause and not significant for other types of failures.
- Mean repair times vary widely across systems, ranging from one hour to more than a day. Repair times depend mostly on the type of the system, and are relatively insensitive to the size of a system.
- Repair times change significantly over the lifetime of a system. Both mean and median repair times drop by more than a third after the first year in operation. This might indicate that during the first year of operation, system administrators get significantly better at quickly identifying the root cause of a problem and fixing it.
- Repair times are extremely variable, even within one system, and are much better modeled by a lognormal distribution than an exponential distribution.

We hope that our first step in analyzing the wealth of information provided by the data, together with the public release of the raw data [1], will spark interesting future work.

Acknowledgments

The authors want to thank Gary Grider, Laura Davey, and the Computing, Communications, and Networking Division at LANL for their efforts in collecting the data and clearing it for public release. They thank the people who were involved in collecting and sharing with us failure data

for system X, but would like to remain anonymous. They also thank Roy Maxion, Priya Narasimhan, and the participants of the ISSRE'05 "Workshop on dependability benchmarking" for their many comments and questions. Finally, they thank the members of the PDL Consortium (including APC, EMC, Equallogic, Hewlett-Packard, Hitachi, IBM, Intel, Microsoft, Network Appliance, Oracle, Panasas, Seagate, and Sun) for their interest and support and the Petascale Data Storage Institute (PDSI) for providing funding for this work.

REFERENCES

[1] The raw data and more information is available at the following two URLs: http://www.pdl.cmu.edu/FailureData/ and http://www.lanl.gov/projects/computerscience/data/, 2006.
[2] X. Castillo and D. Siewiorek, "Workload, Performance, and Reliability of Digital Computing Systems," *Proc. Int'l Symp. Fault-Tolerant Computing (FTCS-11)*, 1981.
[3] J. Gray, "Why do Computers Stop and What Can be Done About It," *Proc. Fifth Symp. Reliability in Distributed Software and Database Systems*, 1986.
[4] J. Gray, "A Census of Tandem System Availability Between 1985 and 1990," *IEEE Trans. Reliability*, vol. 39, no. 4, pp. 409-418, Oct. 1990.
[5] T. Heath, R.P. Martin, and T.D. Nguyen, "Improving Cluster Availability Using Workstation Validation," *Proc. Assoc. Computing Machinery SIGMETRICS*, 2002.
[6] R.K. Iyer, D.J. Rossetti, and M.C. Hsueh, "Measurement and Modeling of Computer Reliability as Affected by System Activity," *ACM Trans. Computer Systems*, vol. 4, no. 3, 1986.
[7] M. Kalyanakrishnam, Z. Kalbarczyk, and R. Iyer, "Failure Data Analysis of a LAN of Windows NT based computers," *Proc. Symp. Reliability in Distributed Software (SRDS)-18*, 1999.
[8] G.P. Kavanaugh and W.H. Sanders, "Performance Analysis of Two Time-Based Coordinated Checkpointing Protocols," *Proc. Pacific Rim Int'l Symp. Fault-Tolerant Systems*, 1997.
[9] T.-T.Y. Lin and D.P. Siewiorek, "Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis," *IEEE Trans. on Reliability*, vol. 39, no. 4, pp. 419-432, Oct. 1990.
[10] D. Long, A. Muir, and R. Golding, "A Longitudinal Survey of Internet Host Reliability," *Proc. Symp. Reliability in Distributed Software (SRDS)-14*, 1995.
[11] J. Meyer and L. Wei, "Analysis of Workload Influence on Dependability," *Proc. Int'l Symp. Fault-Tolerant Computing (FTCS)*, 1988.
[12] B. Mullen and D.R., "Lifecycle Analysis Using Software Defects Per Million (SWDPM)," *Proc. 16th int'l Symp. Software Reliability (ISSRE'05)*, 2005.
[13] B. Murphy and T. Gent, "Measuring System and Software Reliability Using an Automated Data Collection Process," *Quality and Reliability Eng. Int'l*, vol. 11, no. 5, 1995.
[14] S. Nath, H. Yu, P.B. Gibbons, and S. Seshan, "Subtleties in Tolerating Correlated Failures," *Proc. Symp. Networked Systems Design and Implementation (NSDI'06)*, 2006.
[15] D. Nurmi, J. Brevik, and R. Wolski, "Modeling Machine Availability in Enterprise and Wide Area Distributed Computing Environments," *Proc. European Conf. Parallel Computing (Euro-Par '05)*, 2005.
[16] D.L. Oppenheimer, A. Ganapathi, and D.A. Patterson, "Why do Internet Services Fail, and What Can be Done About It?" *Proc. USENIX Symp. Internet Technologies and Systems*, 2003.
[17] J.S. Plank and W.R. Elwasif, "Experimental Assessment of Workstation Failures and Their Impact on Checkpointing Systems," *Proc. Int'l Symp. Fault-Tolerant Computing (FTCS '98)*, 1998.
[18] S.M. Ross, *Introduction to Probability Models*, Academic Press, 1997.
[19] R.K. Sahoo, A. Sivasubramaniam, M.S. Squillante, and Y. Zhang, "Failure Data Analysis of A Large-Scale Heterogeneous Server Environment," *Proc. Dependable Systems and Networks (DSN '04)*, 2004.
[20] B. Schroeder and G.A. Gibson, "A Large Scale Study of Failures in High-Performance-Computing Systems," *Proc. Dependable Systems and Networks (DSN '06)*, 2006.
[21] B. Schroeder and G.A. Gibson, "Disk Failures in the Real World: What Does An MTTF of 1,000,000 Hours Mean to You?" *Proc. Fifth Usenix Conf. File and Storage Technologies (FAST '07)*, 2007.
[22] D. Tang, R.K. Iyer, and S.S. Subramani, "Failure Analysis and Modelling of A VAX Cluster System," *Proc. Int'l Symp. Fault-Tolerant Computing (FTCS)*, 1990.
[23] T. Tannenbaum and M. Litzkow, "The Condor Distributed Processing System," *Dr. Dobbs J.*, 1995.
[24] N.H. Vaidya, "A Case For Two-Level Distributed Recovery Schemes," *Proc. ACM SIGMETRICS*, 1995.
[25] W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson, "Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," *IEEE/ACM Trans. Networking*, vol. 5, no. 1, pp. 71-86, 1997.
[26] K.F. Wong and M. Franklin, "Checkpointing in Distributed Computing Systems," *J. Parallel and Distributed Computing*, vol. 35, no. 1, pp. 67-75, May 1996.
[27] J. Xu, Z. Kalbarczyk, and R.K. Iyer, "Networked Windows NT System Field Failure Data Analysis," *Proc. 1999 Pacific Rim Int'l Symp. Dependable Computing*, 1999.
[28] Y. Zhang, M.S. Squillante, A. Sivasubramaniam, and R.K. Sahoo, "Performance Implications of Failures in Large-Scale Cluster Scheduling," *Proc. 10th Workshop Job Scheduling Strategies for Parallel Processing*, 2004.

Bianca Schroeder received the doctorate degree from the Computer Science Department, Carnegie Mellon University in 2005 under the direction of Mor Harchol-Balter. She is currently an assistant professor in the Computer Science Department, University of Toronto (UofT). Before joining UofT, she was a postdoc for two years at Carnegie Mellon University working with Garth Gibson. She is a two-time winner of the IBM PhD fellowship and has won three best paper awards. Her recent work on system reliability has been featured in articles at a number of news sites, including Computerworld, Slashdot, PCWorld, StorageMojo, and eWEEK.

Garth A. Gibson received the PhD degree in computer science from the University of California at Berkeley in 1991. He is a currently a professor of computer science and electrical and computer engineering at Carnegie Mellon University (CMU) and co-founder and chief technology officer at Panasas, Inc. While at Berkeley, he did the groundwork research and co-wrote the seminal paper on redundant arrays of inexpensive disks (RAID). Joining CMU's faculty in 1991, he founded CMU's Parallel Data Laboratory (www.pdl.cmu.edu), academia's premiere storage systems research center, and co-led the network-attached secure disks (NASD) research project that became the basis of the T10 (SCSI) object-based storage devices (OSD) command set for storage. At Panasas (www.panasas.com), he led the development of the ActiveScale storage cluster in use in government and commercial high-performance computing sites, including the world's first petaflop computer, Roadrunner, at Los Alamos National Laboratory. Panasas products provide scalable performance using a simply managed, blade server platform. Through Panasas, he co-instigated the IETF's emerging open standard for parallelism in the next generation of the network file system (NFSv4.1). He is also the principal investigator of the Department of Energy's Petascale Data Storage Institute (www.pdsi-scidac.org) in the Scientific Discovery through Advanced Computing program and co-director of the Institute for Reliable High Performance Information Technology, a joint effort with Los Alamos. Gibson has been on a variety of academic and industrial service committees including the Technical Council of the Storage Networking Industry Association and the program and steering committee of the USENIX Conference on File and Storage Technologies (FAST). He received the 1999 IEEE Reynold B. Johnson Information Storage Award for outstanding contributions in the field of information storage. He is a member of the IEEE.