A large-scale study of failures in high-performance computing systems

Bianca Schroeder Garth A. Gibson
Computer Science Department, Carnegie Mellon University
Pittsburgh, PA 15217, USA
{bianca,garth}@cs.cmu.edu

Abstract

Designing highly dependable systems requires a good understanding of failure characteristics. Unfortunately, little raw data on failures in large IT installations is publicly available. This paper analyzes failure data recently made publicy available by one of the largest high-performance computing sites. The data has been collected over the past 9 years at Los Alamos National Laboratory and includes 23000 failures recorded on more than 20 different systems, mostly large clusters of SMP and NUMA nodes. 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.

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, 25]. The design and analysis of checkpoint strategies relies on certain statistical properties of failures [8, 21, 23]. 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 sensitive or classified. Existing studies of failures are often based on only a few months of data, covering typically only a few hundred failures [19, 24, 16, 18, 15, 7]. Many of the commonly cited studies on failure analysis stem from the late 80's and early 90's, when computer systems where significantly different from today [3, 4, 6, 13, 19, 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 accompanies the public release of a large set of failure data [1]. The data was collected over the past 9 years at Los Alamos National Laboratory (LANL) and covers 22 high-performance computing (HPC) systems, including a total of 4750 machines and 24101 processors. The data contains an entry for any failure that occurred during the 9-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. 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 it spans, and the number of systems and processors it covers.

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 data. We first describe the environment the data comes 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 concludes.

2 Description of the data and environment

2.1 The systems

The data spans 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 NUMA (Non-Uniform-Memory-Access) nodes, or 2-way and 4-way SMP (Symmetric-Multi-Processing) nodes. In total the systems include 4750 nodes and 24101 processors. Table 1 gives an overview of the 22 systems.

(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	1024	4	12/01 – now	16	2
	6	128	512	4	09/01 – 01/02	16	2
	7	1024	4096	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	1024	4096	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	1024	2	09/03 – now	4	1
				2	03/05 – 06/05	4	1
G	19	16	2048	128	12/96 – 09/02	32	4
				128	12/96 – 09/02	64	4
	20	49	6152	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	1024	0

Table 1. *Overview of systems. Systems 1–18 are SMP-based, and systems 19–22 are NUMA-based.*

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 does 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 1024 and the number of processors ranging from 4 to 6152. 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 vs 16 GB).

2.2 The workloads

Most workloads are large-scale long-running 3D scientific simulations, e.g. for nuclear stockpile stewardship. 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.

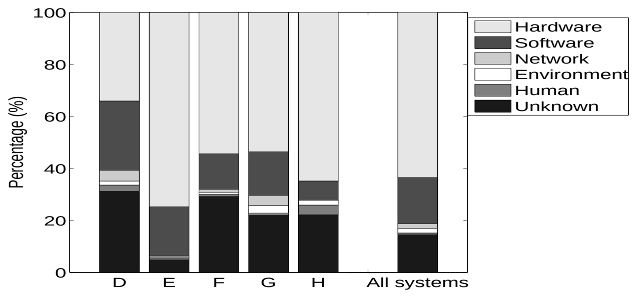
At LANL, failure tolerance is frequently implemented through periodic checkpointing. When a node fails, the job(s) running on it is stopped and restarted on a different set of nodes, either starting from the most recent checkpoint or from scratch if no checkpoint exists.

2.3 Data collection

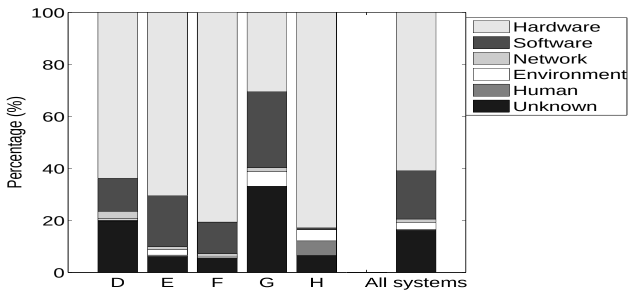
The data is based on a "remedy" database created at LANL in June 1996. At that time, LANL introduced a site-wide policy that requires system administors to enter a description of every failure they take care of into the remedy database. Consequentially, the database contains a record for every failure 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 fail-



(a)



(b)

Figure 1. *The breakdown of failures into root causes (a) and the breakdown of downtime into root causes (b). Each graph shows the breakdown for systems of type D, E, F, G, and H and aggregate statistics across all systems (A–H).*

ure 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 was collected. First, this data is 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 was 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 don't 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.

3 Methodology

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 distribution, 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 six high-level root cause categories: human, environment, network, software, hardware, and unknown. Figure 1(a) shows the percentage of failures in each of the six categories. The right-most bar describes the breakdown across all failure records in the data set. Each of the five bars to the left presents the breakdown across all failure records for systems of a particular hardware type[2].

Figure 1 indicates that while the basic trends are similar across system types, the actual breakdown varies. Hardware

[1]We also considered the Pareto distribution[22, 15], but didn't find it to be a better fit than any of the four standard distributions

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

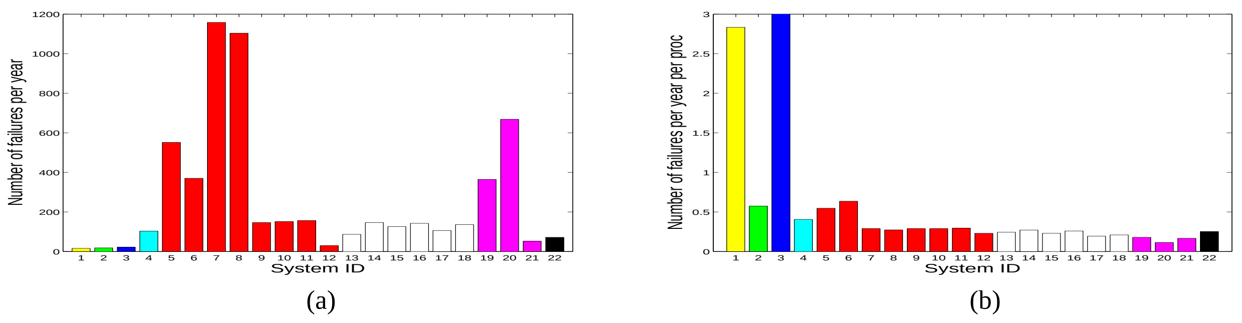


Figure 2. *(a) Average number of failures for each system per year. (b) Average number of failures for each system per year normalized by number of processors in the system. Systems with the same hardware type have the same color.*

is the single largest component, with the actual percentage ranging from 30% to more than 60%. Software is the second largest contributor, with percentages ranging from 5% to 24%. Type D systems differ most from the other systems, in that hardware and software are almost equally frequent.

It is important to note that in most systems the root cause remained undetermined for 20–30% of the failures (except for type E systems, where less than 5% of root causes are unknown). Since in all systems 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.

We also study how much each root cause contributes to the total downtime. Figure 1(b) 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% 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% of downtime with unknown 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 just started to systematically record failure data. As a result, initially the fraction of failures with unknown root causes was high ($> 90\%$), but dropped to less than 10% within 2 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 collected the data. The public release of the data [1] includes a complete FAQ of questions we asked LANL in the process of our work.

In addition to the five high-level root cause categories, we also looked at the more detailed root cause information. We find that in all systems memory related failures make up a significant portion of all failures. For all systems, more than 10% of all failures (not only hardware failures) were due to memory, and in systems F and H memory caused even more than 25% of all failures. Memory was the single most common ”low-level” root cause for all systems, except for system E. System E experienced a very high percentage (more than 50%) of CPU related failures, due to a design flaw in the type E CPU.

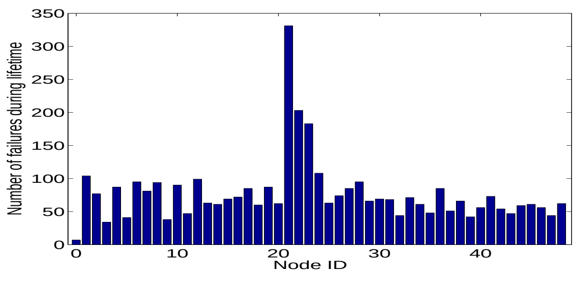
The detailed breakdown for software related failures varies more across systems. For system F, the most common software failure was related to the parallel file system, for system H to the scheduler software and for system E to the operating system. For system D and G, a large portion of the software failures were not specified further.

5 Analysis of failure rates

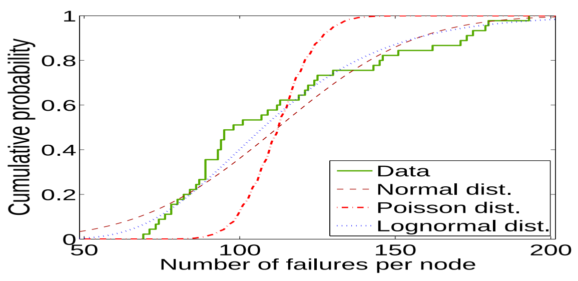
5.1 Failure rate as a function of system and node

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.

Figure 2(a) 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 1159 failures per year for system



(a)



(b)

Figure 3. *(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.*

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. Figure 2(b), 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 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 128–1024 nodes. The same holds for type F systems (systems 13–18), which vary in size from 128–512 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. Figure 3(a) 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 experienced a significantly higher number of failures than the other nodes. While nodes 21–23 make up only 6% of all nodes, they account for 20% 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 Figure 3(a) 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. Figure 3(b) 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 has 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. Figure 4 shows a representative example for each shape.

Figure 4(a) shows the number of failures per month for system 5, 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 systems of type E and F.

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

[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).

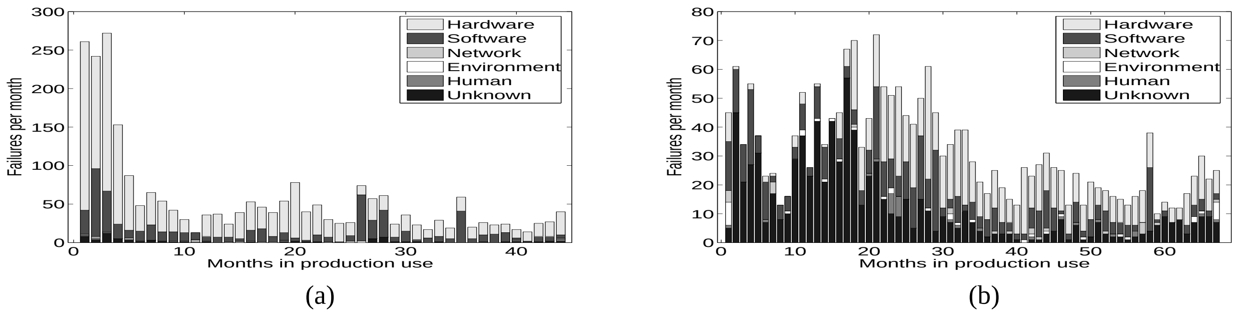


Figure 4. *Two representative examples for how the failure rate changes as a function of system age (in months). The curve on the left corresponds to system 5 which is representative for systems of type E and F. The curve on the right corresponds to system 19 which is representative for systems of type D and G.*

in the production workloads.

The curve in Figure 4(b) 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 type 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 2 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 wouldn't 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 type 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 Figure 4(a). Second, the curve of system 21, which was introduced 2 years after the other systems of type G, is much closer to Figure 4(a).

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. Figure 5 categorizes all failures in the 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

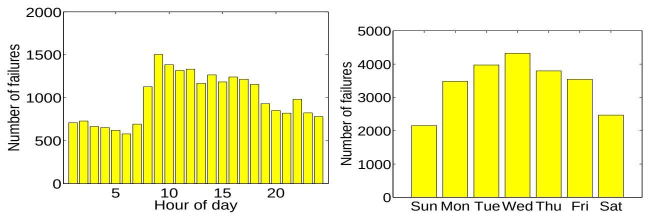


Figure 5. *Number of failures by hour of the day (left) and the day of the week (right).*

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 is lower during the night and on the weekend.

Another possible explanation for the observations in Figure 5 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. We take two different views of the failure process: (i) the view as seen by an individual node, i.e. we study the time between failures that affect only this particular node; (ii) 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 (Figure 4), the time between failures also varies. We therefore analyze the time between failures separately for the early

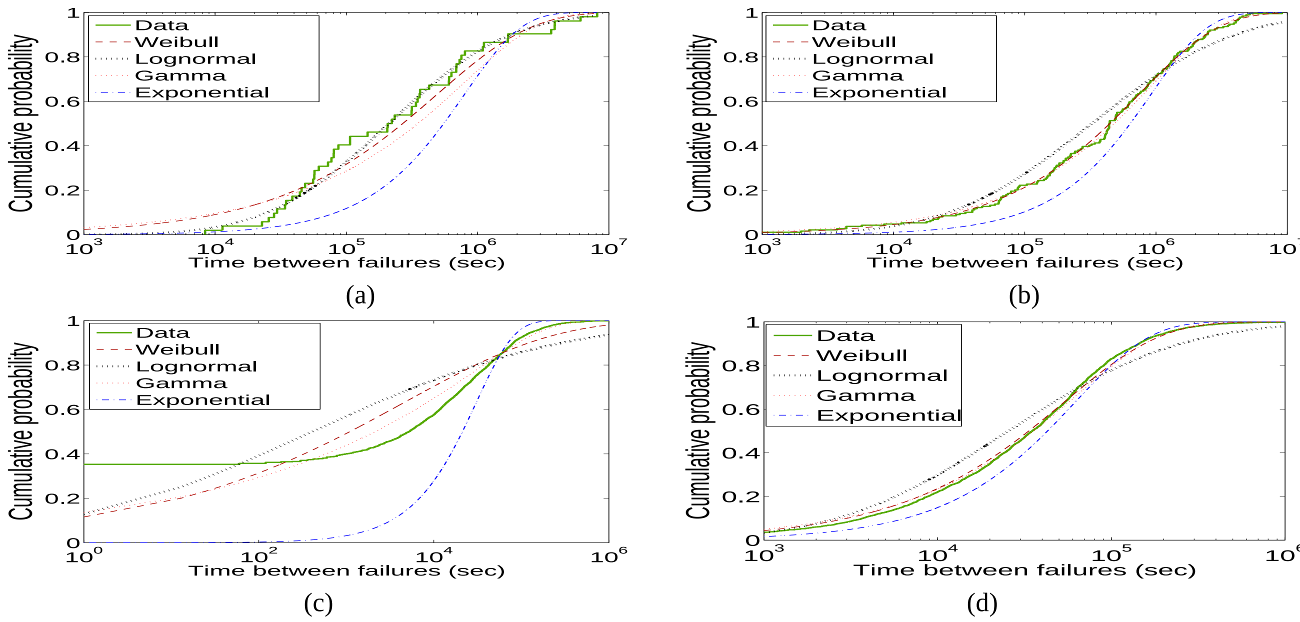


Figure 6. *Empirical CDF for inter-arrival times of failures on node 22 in system 20 early in production (a) and late in production (b). Empirical CDF for inter-arrival times of failures for the system wide view of failures in system 20 early in production (c) and late in production (d).*

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. Figure 6(a) and (b) 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.

For failure interarrival distributions, it is useful to know how the time since the last failure influences the expected time until the next failure. This notion is captured by a distribution's hazard rate function. 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. Figure 6(b) is well fit by a Weibull distribution with shape parameter 0.7, indicating 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.

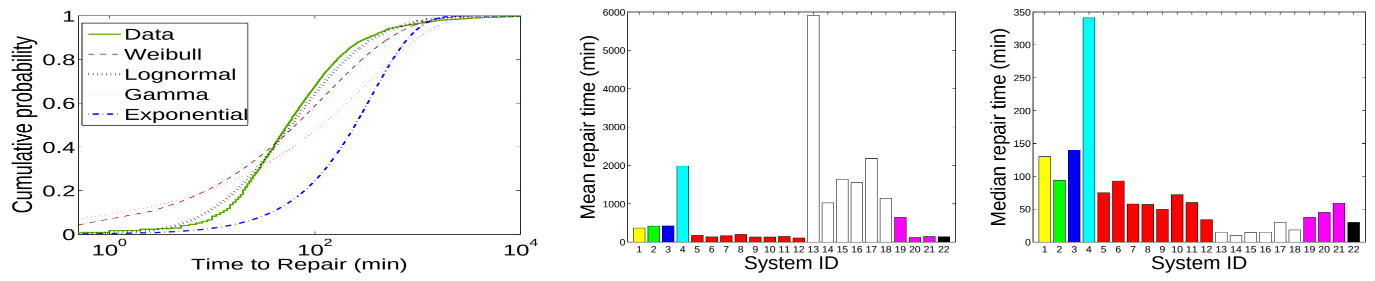
During years 1996-1999 the empirical distribution of the time between failures at node 22 looks quite different (Figure 6(a)) from the 2000-2005 period. During this time the best fit is provided by the lognormal distribution, followed by the Weibull and the gamma distribution. The exponential distribution is an even poorer fit during the second half of the node's lifetime. The reason lies in the higher variability of the time between failures with a C^2 of 3.9. This high variability might not be surprising given the variability in monthly failure rates we observed in Figure 4 for systems of this type during this time period.

Next we move to the system wide view of the failures in system 20, shown in Figure 6(c) and (d). The basic trend for 2000-05 (Figure 6(d)) 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. Again the hazard rate function is decreasing (Weibull shape parameter of 0.78).

The system wide view during years 1996–1999 (Figure 6(c)) 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 ($> 30\%$) of inter-arrival 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.

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



(a) (b) (c)

Figure 7. *(a) Empirical CDF of repair times. (b) Mean repair time and (c) median repair time for each system.*

	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

Table 2. *Statistical properties of time to repair as a function of the root cause of the failure.*

times. We then study the statistical properties of repair times, including their distribution and variability.

Table 2 shows the median and mean of time to repair as a function of the root cause, and as an aggregate across all failure records. 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 3 hours for failures caused by human error, to nearly 10 hours for failures due to environmental problems. The mean time to repair for the other root cause categories varies between 4 and 6 hours. The mean repair time across all failures (independent of root cause) is close to 6 hours. The reason is that it's dominated by hardware and software failures which are the most frequent types of failures and exhibit mean repair times around 6 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 4 times lower than the mean. This high variability is also reflected in extremely high C^2 values (see bottom row of Table 2).

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.

Figure 7(a) 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.

Finally, we consider how repair times vary across systems. Figure 7(b) and (c) 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 type,

Figure 7(b) and (c) also indicate that system size is not a significant factor in repair time. For example, type E systems range from 128 to 1024 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 is less variable than that across all systems, which results in an improved (albeit still sub-optimal) exponential fit[5].

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 failure or error records in the data set. Table 3 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% [3, 13, 16] to 50% [4, 7]. Hardware is reported to make up 10-30% of all failures

[5]Graphs omitted for lack of space.

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
[19]	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
[18]	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
[24]	1999	4 months	503 nodes in corporate envr.	Error logs	2127	TBF
[15]	2005	6–8 weeks	300 university cluster and Condor[20] 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

Table 3. *Overview of related studies*

[4, 13, 16, 7]. Environment problems are reported to account for around 5% [4]. Network problems are reported to make up between 20% [16] and 40% [7]. Gray [4] reports 10-15% of problems due to human error, while Oppenheimer et al. [16] report 14-30%. 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% 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 [18, 19, 5, 24, 15, 10]. Four of the studies use distribution fitting and find the Weibull distribution to be a good fit [5, 24, 9, 15], which agrees with our results. Several studies also looked at the hazard rate function, but come to different conclusions. Some of them [5, 24, 9, 15] find decreasing hazard rates (Weibull shape parameter < 0.5). Others find that hazard rates are flat [19], or increasing [18]. We find decreasing hazard rates with Weibull shape parameter of 0.7–0.8.

Three studies [2, 6, 18] report correlations between workload and failure rate. Sahoo [18] 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. [18] 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% of the nodes in a machine room experience almost 70% 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 Figure 4).

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 don't attempt to fit other distributions to the data.

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.

8 Summary

Many researchers have pointed out the importance of analyzing failure data and the need for a public failure data repository [16]. In this paper we study a large set of failure data that was collected over the past decade at a high-performance computing site and has recently been made publicly available [1]. We hope that this data might serve as a first step towards 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 1000 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, 18].
- 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, 24, 9, 15, 18]. 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).
- Mean repair times vary widely across systems, ranging from 1 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 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.

9 Acknowledgments

We 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. We also thank Roy Maxion, Priya Narasimhan, and the participants of the ISSRE'05 "Workshop on dependability benchmarking" for their many comments and questions. Finally, we 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.

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. In *FTCS-11*, 1981.

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

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

[5] T. Heath, R. P. Martin, and T. D. Nguyen. Improving cluster availability using workstation validation. In *Proc. of ACM 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. Comput. Syst.*, 4(3), 1986.

[7] M. Kalyanakrishnam, Z. Kalbarczyk, and R. Iyer. Failure data analysis of a LAN of Windows NT based computers. In *SRDS-18*, 1999.

[8] G. P. Kavanaugh and W. H. Sanders. Performance analysis of two time-based coordinated checkpointing protocols. In *Proc. Pacific Rim Int. Symp. on 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*, 39, 1990.

[10] D. Long, A. Muir, and R. Golding. A longitudinal survey of internet host reliability. In *SRDS-14*, 1995.

[11] J. Meyer and L. Wei. Analysis of workload influence on dependability. In *FTCS*, 1988.

[12] B. Mullen and D. R. Lifecycle analysis using software defects per million (SWDPM). In *16th international symposium on 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 Engineering International*, 11(5), 1995.

[14] S. Nath, H. Yu, P. B. Gibbons, and S. Seshan. Subtleties in tolerating correlated failures. In *Proc. of the Symp. on 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. In *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? In *USENIX Symp. on Internet Technologies and Systems*, 2003.

[17] J. S. Plank and W. R. Elwasif. Experimental assessment of workstation failures and their impact on checkpointing systems. In *FTCS'98*, 1998.

[18] 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 DSN'04*, 2004.

[19] D. Tang, R. K. Iyer, and S. S. Subramani. Failure analysis and modelling of a VAX cluster system. In *FTCS*, 1990.

[20] T. Tannenbaum and M. Litzkow. The condor distributed processing system. *Dr. Dobbs Journal*, 1995.

[21] N. H. Vaidya. A case for two-level distributed recovery schemes. In *Proc. of ACM SIGMETRICS*, 1995.

[22] 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. on Networking*, 5(1):71–86, 1997.

[23] K. F. Wong and M. Franklin. Checkpointing in distributed computing systems. *J. Par. Distrib. Comput.*, 35(1), 1996.

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

[25] Y. Zhang, M. S. Squillante, A. Sivasubramaniam, and R. K. Sahoo. Performance implications of failures in large-scale cluster scheduling. In *Proc. 10th Workshop on Job Scheduling Strategies for Parallel Processing*, 2004.