Failure Data Analysis of a Large-Scale Heterogeneous Server Environment

Ramendra K. Sahoo
Exploratory Server Systems Dept.
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA

Anand Sivasubramaniam
Dept. of Computer Science & Engg.
Pennsylvania State University
University Park, PA 16802, USA

Mark S. Squillante
Mathematical Sciences Dept.
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA

Yanyong Zhang
Dept. of Electrical & Computer Engg.
Rutgers University
Piscataway, NJ 08854, USA

Abstract

The growing complexity of hardware and software mandates the recognition of fault occurrence in system deployment and management. While there are several techniques to prevent and/or handle faults, there continues to be a growing need for an in-depth understanding of system errors and failures and their empirical and statistical properties. This understanding can help evaluate the effectiveness of different techniques for improving system availability, in addition to developing new solutions. In this paper, we analyze the empirical and statistical properties of system errors and failures from a network of nearly 400 heterogeneous servers running a diverse workload over a year. While improvements in system robustness continue to limit the number of actual failures to a very small fraction of the recorded errors, the failure rates are significant and highly variable. Our results also show that the system error and failure patterns are comprised of time-varying behavior containing long stationary intervals. These stationary intervals exhibit various strong correlation structures and periodic patterns, which impact performance but also can be exploited to address such performance issues.

1. Introduction

Our growing reliance on computing and information processing services mandates deploying systems that can not only meet the performance demands imposed on such systems, but are also available when needed. Several technological factors are accentuating the problem of system failures, which are highly undesirable since these systems could be servicing the needs of hundreds of users. At the same time, solutions for this problem need to keep the high costs of system maintenance personnel in mind, which is growing to be a much more important factor in Total Cost of Ownership (TCO). A deep understanding of the occurrence of failures in real environments can be useful in several ways towards enhancing overall system availability. It can provide realistic data when evaluating proposed solutions, together with developing strategies for pro-active prediction and remedies of faults ahead of their occurrence. Towards this goal, in this paper we conduct a detailed empirical and statistical analysis of system errors and failures from a production environment of nearly 400 server machines with data collected over a year.

As our growing needs for performance continue to fuel the development of high performance systems/servers, there are several associated technological factors to keep in mind:

- Denser integration of semiconductor circuits, though preferable for performance, makes them more susceptible to strikes by alpha particles and cosmic rays [20]. At the same time, there is an increasing tendency to lower operating voltages in order to reduce power consumption. Such reduction in voltage levels can increase the likelihood of bit-flips when circuits are bombarded by cosmic rays and other particles, leading to transient errors. While memory structures are typically the target for protection against errors using informational redundancy, more recent studies [14] have pointed out that the error rates in combinational circuits are likely to surpass those of memory cells in the next decade.
- At the macro granularity, we have dense blade-systems being packed in a rack as a cluster. With a high load imposed on these dense systems – both on the CPUs and on the disks – heat dissipation becomes a very important concern, potentially leading to thermal instability that can cause system/node breakdowns [1].

0-7695-2052-9/04 $ 20.00 © 2004 IEEE

- We find system software and applications becoming more complex. Such complexity makes them more prone to bugs and other software failures [15, 9, 18] (e.g., memory leaks, state corruption, etc.). These bugs/failures can cause system crashes.

- Networking (whether it be the Internet, or a local/system/storage area network) has made it convenient to deploy systems that are inherently parallel in nature (whether it be functional – different systems performing different operations – or data parallelism – different systems performing same operations but on different pieces of data). This can not only be performance-efficient, but can also make it easier to write and deploy distributed programs/systems. However, the growing reliance on each other make nodes within a parallel/distributed system more susceptible to another's failures/errors [19].

All these factors point to the increasing occurrence of system failures in the future. Rather than treat them as an exception, system design needs to recognize fault occurrence, and manage the resources effectively so as to hide their impact from the end users. One would ideally like to achieve the performance of a system without any failures. Even if this is difficult to attain, there should be at most a "graceful degradation" in performance under the presence of failures.

There are several broad solution strategies for addressing this issue:

- We could implement designs that recognize the occurrence of failures when the system/application is deployed initially and provision sufficient redundancy to avoid problems when failures occur. For instance, one could provision redundancy in the I/O subsystem (e.g., RAID), duplicate file servers themselves, etc. Though this may not be the most effective use of the additional hardware/software, which may have been used to improve performance in the first place, the diminishing fraction of the hardware/software procurement contribution to the TCO may motivate the use of this strategy in some environments.

- Instead of provisioning redundancy at deployment, one could anticipate failures just ahead of their occurrence and perform pro-active counter-measures. For instance, it has been suggested that one should perform pro-active shutdown/rejuvenation [18] to avoid catastrophic consequences due to different software failures occurring due to system aging. Similarly, finding alternate servers (or alternate routes to the same server), by-passing any non-important steps in the execution, etc., may be done when failures are anticipated.

- Instead of anticipating failures and taking pro-active measures, the option may be to deal with failures when they actually occur, e.g., replacing cluster nodes/disks when they fail, reconfiguring routers, etc. Some of this may be automated, though a human may be needed for a lot of other actions.

Each of the above techniques has its pros and cons, and their extensive trade-offs may need to be analyzed in detail to pick the best option (or a combination) for the environment being studied. Our point here is not to favor any one over another. Rather, we note that across these techniques there is an important need to gain a detailed understanding of system errors and failures and their underlying properties. For instance, the choice of using redundancy at initial deployment would depend largely on the frequency of failures – higher frequency favors the choice of this technique. The effectiveness of pro-active action depends significantly on how accurately we can predict the occurrence of failures. Being very conservative when we cannot predict failures accurately can affect performance. In the strategy of taking actions only when failures occur, understanding failure properties can help prepare personnel appropriately (help select personnel based on their expertise), order spare parts, reduce the down-time by isolating the problem and proposing remedies, etc. Finally, across these techniques, a characterization of system errors and failures can serve as a valuable workload to study the effectiveness of any schemes that may be developed. All these serve as motivating reasons for studying the errors and failures in real production systems and characterizing their behavior in detail.

Recognizing the importance of studying system failures, there have been several attempts at a detailed characterization of data from real-world systems as will be summarized in the next section. Many of these studies [16, 17, 19, 3] have been in the 1980s and early 1990s, where the hardware and software were significantly different. Today's systems pose a much higher level of sophistication in the kinds of applications that they run, and in the complexity of the system software and hardware. A more closely related study is the one presented in [19] where event logs from a networked system of around 500 Windows NT server systems is studied. Their study uses a production environment from a business enterprise where the applications are mainly commercial, in addition to services such as file and email service. Beyond including such commercial services, the networked system of servers that we examine is much more diverse, containing heterogeneous hardware ranging from single CPU servers to 2/4/8/12-way SMPs, and even clusters, and the workload includes several long-running scientific applications as well.

We have collected extensive error logs from a networked environment of heterogeneous servers at the IBM Thomas

J. Watson Research Center consisting of close to 400 machines (many having multiple processors) over a period of close to 500 days that provides information about both non-critical events (which may simply be alarms) and critical events (which cause node failures). We conduct detailed empirical and statistical analysis of these event logs. The results of our analysis demonstrate that, even though there continue to be considerable improvements in system robustness to limit the number of actual failures to a very small fraction of the recorded errors, the failure rates are quite significant and highly variable. Our results further show that the system error and failure patterns are comprised of non-stationary behavior containing long stationary intervals many spanning more than a day. These stationary intervals exhibit various forms of strong correlation structures, including possible long-range dependence, together with significant periodic behavior and considerable cross-correlation between less-serious errors and more-serious errors plus failures. Moreover, the failures are not uniformly distributed among the nodes and a small fraction of the nodes incur most of the failures, with less than 4% of the nodes experiencing almost 70% of the failures. This behavior has also been observed in other studies [16, 8, 19] that have attempted to characterize the failure properties of real systems. The failures at the nodes incurring the most failures also have a strong temporal correlation with time of day at the hourly level and these temporal correlation patterns vary over time with such behavior differing among the individual nodes.

The rest of this paper is organized as follows. The next section provides a brief summary of work related to this study. An explanation of the system log collection and the associated analyses are given in Sections 3 and 4, respectively. Finally, Section 5 summarizes the results of this study.

2. Related Work

The impact of failures on dependability and availability of distributed server systems has received much attention over the last decade. Tang [16, 17] studied the error/failure log collected from a VAXcluster system consisting of seven machines and four storage controllers. They found that 98.8% of the errors are recovered, and 46% of the failures are caused by external errors. Using a semi-Markov failure model, they further pointed out that failure distributions on different machines are correlated rather than independent. Xu [19] performed a study of error logs collected from a heterogeneous distributed system consisting of 503 PC servers. They showed that failures on a machine tend to occur in bursts, possibly because common solutions such as reboots cannot completely remove the problem causing the failure. They also observed a strong indication of error propagation across the network, which leads to the correlation between failures of different nodes. These studies use the empirical (marginal) distribution obtained from their respective datasets to build failure models. Heath [3] collected failure data from three different clustered servers, ranging from 18 workstations to 89 workstations. They assumed the times between failures are independent and identically distributed, and fit the failure data using a Weibull distribution with a shape parameter less than 1. In addition, they used a property of this Weibull distribution – nodes that just failed are more likely to fail again in the near future – to motivate a new resource management strategy.

The impact of workload on both software and hardware failures has also been studied. Castillo [2] considered workload as a factor in predicting software failures. Iyer [6] examined the effect of workload on the reliability of the IBM 3081 operating system. The study validated a load-hazard model. Mourad [11] conducted a study on the IBM MVS/XA operating system and found that error distribution is heavily dependent on the type of system utilization. Meyer [10] looked at the impact of workload on the computer system dependability. Vaidyanathan [18] explained that software related error conditions will accumulate over time which will eventually lead to crashes/failures.

3. System Environment and Failure Data

Environment: Our study is based on events collected from 395 nodes in a machine room at the IBM Thomas J. Watson Research Center that support a broad and diverse user community. Note that we use the term *node* to denote one machine, irrespective of the number of CPUs it may contain. For instance, a 4-way or 8-way SMP is still denoted as 1 node. All of the nodes run (possibly different versions of) the AIX operating system and are connected by a high-speed network. The processors are also heterogeneous across the collection of nodes. The majority of nodes are primarily used for scientific applications, with a few of the nodes being used to execute commercial applications. Most of the scientific applications are parallel codes using MPI that solve problems related to weather forecasting, speech recognition, etc., while most of the commercial applications are parallel and sequential codes for database, mail serving, and related activities. A description of the workloads running on these 395 nodes is provided in Table 1.

Event Collection Mechanism: We obtained extensive event logs from these 395 nodes over a representative period of 487 days during 2002 and 2003. This was procured using the Reliability Availability and Serviceability (RAS) facility of the AIX operating system. This facility uses the already available mechanism wherein the kernel/applications log their errors in the special file $/dev/error$. A daemon ($errdaemon$) is started during system initialization, which

No. of Nodes	Description
72	Computational Linguistics
48	Computational Physics
50	Libraries/Computing Services
112	Protein Folding
8	DB2 server
2	SAMBA Server
15	DCE/DFS
13	GPFS/GSA File Server
3	Web server,Loadleveler/scheduler
20	Misc. (incl. license server, mail server, etc.)
4	ADSM/Backup system
48	Weather modeling

Table 1. Node breakup for workloads.

continuously monitors this special file for any events of interest. The label of each new entry in this file is compared against a database of error record templates. If there is a match, then additional information about the system environment and/or hardware status is added, before the entry is posted to an error log. The default temporal granularity for logging in AIX is 1 minute, and typically system configuration parameters such as buffer sizes are set based on this default to avoid excessive overheads. Though this could be modulated for a finer resolution, we opted against doing so, because: (i) it could affect the performance of many server critical applications, thus interfering with their everyday usage; (ii) it requires changing the resolution on all the nodes which may not be very desirable (on the other hand 1 minute resolution is provided as a default); and (iii) as will be described shortly, we eventually need to filter the data (eliminating multiple reports of the same errors) which uses a resolution of about 5 minutes to remove redundant events within this window, making a resolution of 1 minute sufficient for the logging itself. More details of RAS logging mechanism for AIX are described in [4].

Each entry of the system error log contains information on various aspects of the error. This includes the node number (*Node*) which denotes the location of the error, error identifier (*ID*), time stamp (*Time*), error type (*Type*) which denotes the error severity, error class (*Class*) which denotes the subsystem affected by the error and a short description of the problem (*Description*). AIX error logging provides error templates for 800 different types of errors. These errors can be broadly classified based on the type of error (i.e., the error severity) and affected subsystem (i.e., hardware or software related). Specifically, there are 5 error types: (1) TEMP: Error condition recovered after a number of unsuccessful attempts; (2) PERF: Performance of the device/component has degraded to below an acceptable level; (3) UNKN: Unknown error (Cannot determine the severity); (4) PEND: Loss of availability of a device or component is imminent; and (5) PERM: Permanent Error (Unrecoverable/Most Severe Error). Similarly, there are 4 error classes of interest: (1) Class O: Informational errors only; (2) Class S: Software related errors; (3) Class U: Undetermined errors; and (4) Class H: Hardware related errors. We provide a few representative error examples along with their error types and error classes in Table 2.

Type	Class	Description
PEND	H	CPU Failure Predicted
TEMP	H	Communication Protocol Error
PERF	O	Unable to Allocate Space in Kernel Heap
PEND	S	Potential Data Loss Condition
TEMP	O	Recoverable Software Error
PERM	S	Configuration Failed

Table 2. Examples of Error log entries.

Event Processing and Filtering: The raw event logs collected using the above mechanism include planned and scheduled maintenance, shutdowns and reboots, in addition to other errors in hardware and software. In all, a total of $6,533,152$ raw events were collected from these 395 nodes over a period of 487 days. It is to be noted that this raw data needs to be processed in order to eliminate certain repetitive/redundant events, system outages and scheduled maintenance, so as to be useful in subsequent analysis. Such filtering of raw data is a difficult task, as has also been pointed out in related studies [16]. Our filtering process consists of several iterations of spurious and redundant log removal, using a *time window* defined as *threshold time* (T_{th}). The basic premise is that events that appear within a very short time interval at a node are a result of the same errors [16]. The filtering algorithm records any new event type as a new *event ID* and any new node number as a new *node ID*. It compares the *event ID* and *node ID* at any time T with the *event ID* and *node ID* of all events since time $(T - T_{th})$. If both the IDs are same, the filtering process only records the *event ID* and *node ID* corresponding to time $(T - T_{th})$, discarding the recorded event at time T. If either of the IDs are different, it records the events as two distinct events.

There are trade-offs that we need to consider when choosing T_{th}. On the one hand, a very small value would lessen the chances of removing two entirely different events that could be tagged as duplicates when a larger threshold value is used. At the same time, the smaller threshold would eliminate a much smaller fraction of spurious events. We found that a threshold value of 5 minutes (i.e., $T_{th} = 5$ minutes) did a fairly good job of eliminating spurious events while bringing down the size of the log considerably. Note that this threshold value is also comparable to the values used in related studies [16].

Through this simple filtering process, we were able to eliminate 91.82% of the total raw events, ending up with a total of $534,680$ distinct events, over the period of 487 days. As the result of a major scheduled system maintenance event within this time interval and because our statistical procedure for determining stationarity requires input time-series whose lengths are a power of 2, we identified a representative period of 364 days (2^{19} minutes) from the original dataset and use this refined dataset as the basis for our study. The refined dataset consists of $520,303$ filtered events over the 364 days whose properties are investigated as part of our empirical and statistical data analysis.

Note that not all of these events are actual failures; e.g., some could be informative or warnings as explained in the error classification above. The information in the system error log entries made it possible for us to identify actual failures that caused the corresponding node to go down. Such failures consist of PERM type errors and a small subset of PEND type error entries in the log. We shall henceforth refer to this collection of entries in the log as *failures*. The remaining PEND type log entries together with the TEMP, PERF and UNKN type entries in the system log represent actual *errors* detected and logged by RAS that do not result in a failure. This is either because of some corrective action taken to address the problem or because the problem was transient. Table 3 provides a breakdown of the error and failure counts categorized by the type and class fields in their log entry.

	Class O	Class S	Class U	Class H	Total
TEMP	11420	1000	141705	143841	297966
PERF	1929	0	8410	28828	39167
UNKN	9109	4352	6646	12	20119
PEND	500	156613	5274	0	162387
PERM	0	0	0	664	664
TOTAL	22958	161965	162035	173345	520303
Failures	498	123	0	664	1285

Table 3. Error and failure counts.

4. Data Analysis

In this section we present the results of an analysis of the error and failure data from the large distributed system described in Section 3. Our focus is on a representative interval of the data consisting of a total of 364 days (2^{19} minutes) starting in early 2002 and ending in early 2003. The corresponding datasets are obtained after prefiltering and removal of extraneous events such as scheduled system maintenance, as discussed in Section 3. We first consider the properties of the aggregate system-wide error and failure data, starting with a simple empirical analysis of the raw datasets and then turning to a deeper statistical analysis of the properties of the underlying stochastic processes. We then consider some related properties of such events from the perspective of individual nodes.

4.1. System-Wide Errors and Failures

The time-series datasets of failures and different types of errors contain periods with multiple events every minute and other periods with no events over many consecutive minutes. Hence, our analysis considers two different but related views of each of these time-series datasets, i.e., the number of events per minute, or *rate process*, and the time between events, or *increment process*, for the sequences of failures and different types of errors.

General statistics for the raw rate processes (i.e., number of events of each type, daily average number of events of each type, and the average, variance and maximum of the number of events of each type per minute) and the raw increment processes (i.e., number of minutes spanned by the events of each type, and the average, variance and maximum of the number of minutes between events of each type) are provided in Table 4. We observe that the failures represent a relatively small fraction of the events in the error logs, with the number of TEMP and PEND errors exceeding the number of failures by 2 orders of magnitude while the number of PERF and UNKN errors exceed the number of failures by an order of magnitude. This suggests that a large number of errors may be resolved before an actual failure occurs, which could be due to increased error reporting or improved system robustness or a combination of both. We further observe that the average time between failures is 2 orders of magnitude larger than the average time between all errors (see row 2 of Table 4(b)). The variability of the interfailure times is also relatively large having a coefficient of variation (CV: ratio of standard deviation to mean) that exceeds 2.5, whereas the variability of the intererror times is significantly larger having a CV that exceeds 14. Many of these trends are fairly consistent with previous results in the literature (e.g., [16, 19, 5]), although the fraction of error events that represent actual failures (0.25%) appears to be smaller (possibly reflecting improvements in error reporting and/or system robustness), the rate at which errors and failures occur have increased (possibly because reporting capabilities have improved and because hardware and software have become more complex), and the variance in such error/failure occurrences have also increased.

To gain a better understanding of the properties of these raw failure and error datasets, we next consider the corresponding empirical (marginal) distribution for each of the processes. As a representative sample of our results, Figure 1 presents the empirical distribution for the interfailure times (a) and the empirical distribution for the number of errors per minute (b), where both axes are in log-scale. Observe the non-monotonic characteristics in both processes that suggest the raw datasets may contain mixtures of different statistical behaviors, which is most apparent in the error rate process plots. We further observe from these re-

	TEMP	PERF	UNKN	PEND	All Errors	Failures
Events	297966	39167	20119	162387	519018	1285
Daily Avg	818.39	107.58	55.26	446.01	1425.53	3.53
Average	0.57	0.07	0.04	0.31	0.99	0.002
Variance	12.42	1.03	0.28	7.76	28.25	0.01
Maximum	468	175	99	350	691	16

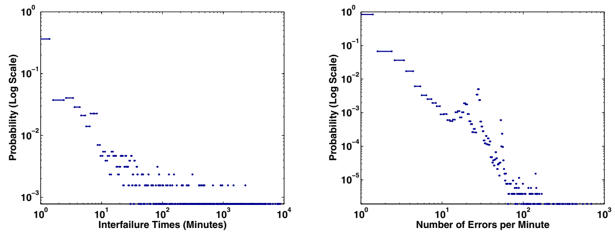
(a)

	TEMP	PERF	UNKN	PEND	All Errors	Failures
Minutes	524229	522279	523415	523822	524229	519486
Average	1.76	13.33	26.02	3.24	1.01	404.27
Variance	412.70	9028.64	18841.8	2821.2	207.2	1045520
Maximum	5794	6267	5917	5971	5792	8631

(b)

Table 4. General statistics: (a) rate processes; (b) increment processes. (Times in minutes.)

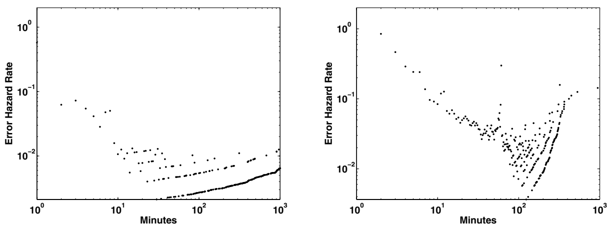
sults: (1) the very rapid initial declines in the distribution functions, where the majority of the probability mass occurs for very small values on the x-axis; and (2) the relatively long tails of the distributions for large values on the x-axis, where there are also significant gaps in the tail distribution with probability mass 0 for relatively large intervals on the x-axis. The decay rates of the tails flatten out for larger values on the x-axis, although the tails are obviously bounded at the maximum values provided in Table 4. On the other hand, it is important to point out that the statistical significance of these long tails in both plots is quite limited, given that the set of points with the smallest positive probability mass represents occurrences of a single event and the set of points with the next smallest positive probability mass represents occurrences of two events. These results on the tail properties of the empirical marginal distributions help to explain the corresponding high variance of the failure and error processes which were previously observed for the CV. While some of these trends are consistent with previous results in the literature [16, 19, 3], the differences in magnitude and other characteristics (such as tail properties and non-monotonicity) noted above are reflective of the differences in our raw failure and error datasets.



(a) (b)

Figure 1. Empirical (marginal) distributions.

In an attempt to explore the empirical properties of these marginal distributions of the raw datasets in more detail, we consider the corresponding empirical hazard rate functions [7] defined as $h(x) = f(x)/(1 - F(x))$ where $f(x)$ is the marginal density function and $F(x)$ is the marginal cumulative distribution function. Figure 2 plots the hazard rate function for the marginal distributions in Figure 1 of the raw interfailure (a) and intererror (b) times. We observe that $h(0) < 1$ because $f(0) < 0.5$ for the interfailure times and that $h(0) > 1$ because $f(0) > 0.5$ for the intererror times. The empirical hazard rate functions of both processes decrease rapidly with increasing x, which is as expected and is also consistent with previous results (e.g., [16, 3]). However, for larger values of x, we observe that $h(x)$ tends to rise with increasing values of x and this is especially sharp for the intererror times. This implies that failures and errors become increasingly likely as the time since the last failure and error increases beyond a certain point, strictly with respect to the marginal distributions of the corresponding raw interfailure and intererror datasets. Such behavior is different from some previous results where the trend in the tail of the empirical hazard rate function is either flat (e.g., [16]) or decreasing (e.g., [3]). We further observe from the results in Figure 2 additional evidence of possible mixtures of different statistical behaviors in the raw error and failure datasets.

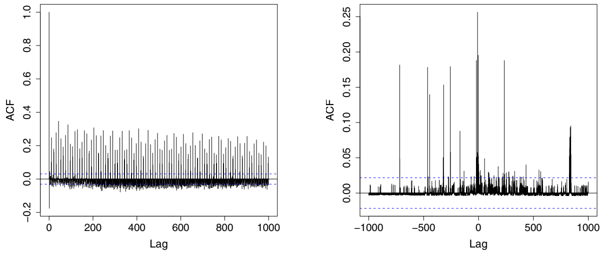


(a) (b)

Figure 2. Hazard rate functions.

The above results present an empirical analysis of the raw datasets used in our study together with a few comparisons against some of the corresponding results in the research literature. In order to gain a deeper understanding of the statistical properties of our time-series datasets and to aid in the development of stochastic failure models based on this data, we turn to consider a more detailed statistical analysis of the time-varying properties and correlation structures of the underlying stochastic error and failure processes. The first important issue of interest concerns the stationarity of these error and failure processes, where a time series is said to be stationary if it is invariant to shifts in time with respect to statistical measures. Motivated by its effective use in a recent Web traffic study [13], we use a specific statistical procedure, called Auto-SLEX [12], to partition the error and failure rate and increment processes into stationary intervals. Auto-SLEX has the advantages that it does not identify changes in the level of a time series and it only identifies changes in the correlation structure, because the partitioning is based on second-order properties of the process and ignores the level information contained in the spectral value at zero frequency; refer to [12, 13] for additional technical details. The corresponding results for error/failure rate and increment processes demonstrate that these processes are clearly nonstationary and that they contain relatively long time intervals which are stationary. In particular, during the entire dataset interval of 2^{19} minutes we find 2 stationary intervals in the error+failure rate process of length 8192 minutes, 5 stationary intervals of length 4096 minutes, 9 stationary intervals of length 2048 minutes, and 19 stationary intervals of length 1024 minutes. It is quite interesting to observe so many long stationary intervals, representing more than 5.6 days, 2.8 days, 34 hours and 17 hours, respectively. Moreover, during the same dataset interval we find a few stationary intervals in the failure increment process that span more than 28 days. Note that these results are very consistent across a wide range of parameter settings for the Auto-SLEX procedure, including its defaults.

Another important statistical issue of interest concerns the correlation structures contained within stationary intervals of the error and failure processes. We therefore consider next the sample autocorrelation function (ACF) [7] for the stationary intervals of the processes of interest. As a representative sample of our results, Figure 3(a) plots the ACF for a stationary interval of the error+failure rate process of length 4096 minutes. These results demonstrate significant correlation structures together with significant periodic behavior on the order of 10–11 minutes. The slow decay of the ACF with increasing lag suggests possible long-range dependence within relatively long stationary intervals of the error+failure rate process. It is interesting to note that very consistent ACF results were obtained for different stationary intervals of this process of length 4096 minutes, including almost identical periodic behavior on the order of 10–11 minutes. To gain a deeper understanding of the strong correlation structures in the error+failure rate process, we partition the set of errors and failures into less-serious errors (TEMP, PERF, UNKN) and more-serious errors (PEND) plus failures, and then compute the ACF for each of these 2 separate time series. Interestingly, while each of these ACFs exhibit considerable correlation structures, the magnitude of this correlation is much smaller than that shown in Figure 3(a) and this correlation dies out fairly quickly with increasing lag. Some forms of periodic behavior also exist in both ACFs, but once again the magnitude is smaller and it dies out more quickly. These results suggest that the strong correlation structures in the error+failure rate process of Figure 3(a) might be due, in part, to the interactions between the statistical properties of the less-serious errors and the more-serious errors plus failures.



(a) (b)

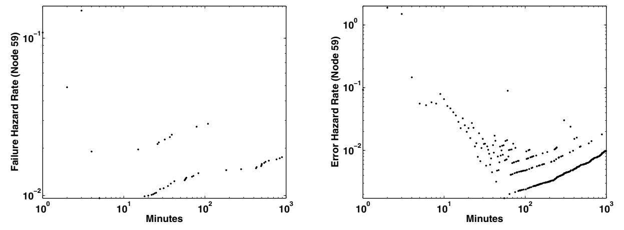
Figure 3. Sample ACF and CCF.

In order to further explore the strong correlation structures of the error+failure rate processes, we next consider the sample cross-correlation function (CCF) [7] between common stationary intervals for the less-serious error rate processes and the more-serious error plus failure rate processes discussed above. Figure 3(b) plots a representative sample of our CCF results for a common stationary interval of length 4096 minutes. Note that the positive-valued lags of the CCF plot show the relationship between less-serious errors and more-serious errors plus failures at lags of k minutes apart, whereas the negative-valued lags show the correlation between less-serious errors and more-serious errors plus failures at lags of $-k$ minutes apart (i.e., the less-serious errors follow the more-serious errors plus failures with a distance of k minutes). We observe from these results that there are significant cross-correlation structures between these two partitions of the error+failure rate process together with considerable periodic behavior. It is interesting to note that fairly consistent CCF results were obtained for different common stationary intervals of the error+failure rate process. These results tend to support the view that the strong correlation structures in the error+failure rate processes might be due, in part, to the statistical interactions between the time-series of less-serious and more serious events, and they provide further insights into the forms of these statistical interactions.

4.2. Per-Node Errors and Failures

System errors and failures most directly impact the nodes where they occur, as this is the level at which system performance is impacted and can be addressed. In this section we analyze some of the properties of the error and failure datasets from the perspective of individual nodes to investigate these issues and to further gain a better understanding of the underlying stochastic error and failure processes.

We first consider the empirical (marginal) distribution for each of the raw failure and error datasets at individual nodes, together with the empirical hazard rate functions for the corresponding marginal distributions of the raw interfailure and intererror times. Upon comparing our results with those in Figure 1, we find that each of the empirical marginal distributions for the individual nodes exhibit characteristics and trends fairly similar to those observed for the corresponding system-wide marginal distributions. A representative sample of the empirical hazard rate functions for these per-node marginal distributions from the perspective of a specific node (ID 59) is provided in Figure 4, both for the empirical distribution of the raw interfailure times (a) and the empirical distribution of the raw intererror times (b). We observe that the hazard rate functions in Figure 4 for an individual node have characteristics and trends similar to those found in the corresponding system-wide hazard rate functions in Figure 2, although there are some considerable differences in magnitude. This suggests that some of the empirical properties associated with the marginal distributions for the system-wide raw failure and error datasets may also hold to some extent for a subset of the individual nodes, especially those which encounter a relatively large number of errors and failures.



(a) (b)

Figure 4. Hazard rate functions for node 59.

We next consider the manner in which the raw failures are distributed among the individual nodes comprising the distributed system. Figure 5 plots the number of failures as a function of the node ID, where the nodes are listed in nonincreasing order with respect to the number of failures. These results clearly illustrate that the raw failures are not uniformly distributed among the set of nodes and that a small fraction of the nodes incur the overwhelming majority of the failures. In fact, less than 4% of the nodes incur almost 70% of the failures. The distribution has a large probability mass for the first 3 nodes, followed by a rapid decrease to a second tier of probability mass, followed by a rapidly decaying tail. Clearly there are strong forms of correlation between certain individual nodes and failures. According to [2, 6, 5, 11], the workload has an important impact on the reliability of the underlying system. Thus, upon examining the workloads running on each node, we find that 3 of the top 5 nodes (IDs 82, 79 and 330) were serving as file servers, while the remaining 2 nodes (IDs 59 and 4) were serving as database servers. This suggests the following possible explanation for the high failure rates on these 5 nodes. Firstly, it seems reasonable to hypothesize that the nodes operating as file/database servers for the entire distributed system will experience a much higher load than the other computing nodes, even though the utilization of each node is not available to us in our study. Secondly, earlier studies such as [16] have observed that most (hardware) failures occur in the I/O subsystem, which is a primary operation of a file/database server.

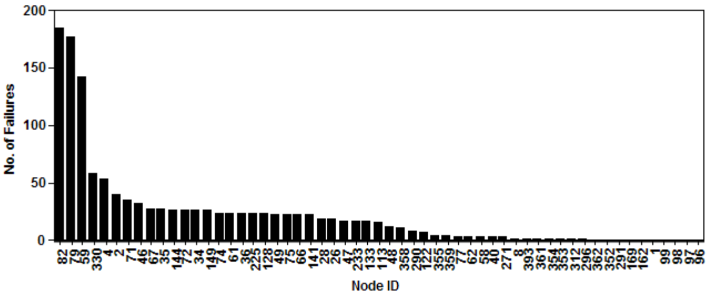


Figure 5. Failures as a function of node ID.

Given the strong concentration of failures on relatively few nodes, we next consider the empirical (marginal) distribution of the time at which failures occur on the 5 nodes with the most failures. Figure 6 plots the corresponding results where the raw failures are aggregated at an hourly level for each of the 24 hours in a day. These results clearly illustrate that the failures at the top 5 nodes also have strong correlations with time of day at the hourly level. Node 82 encounters essentially no failures after 1am until around 6 am when a concentrated probability mass of failures continues to rise over several consecutive hours through noon, and then there is a decreasing trend in the probability mass of failures through midnight. The failures experienced at node 79 are somewhat more persistent and spread out over the 24 hours in a day, with positive failure intensities every hour and with peak failure intensities occurring in the 14th, 1st, 11th and 20th hours. Nodes 79 and 82 incur the largest per-hour failure intensities at hours 14 and 12, respectively. The failures encountered at node 59 are also somewhat persistent with positive failure intensities every hour and they consist of several cycles of varying forms of oscillating failure intensities. Node 330 experiences somewhat fewer cycles of relatively similar forms of oscillating failure intensities throughout the 24 hours in a day, although the variability of these peaks and valleys is considerably smaller than

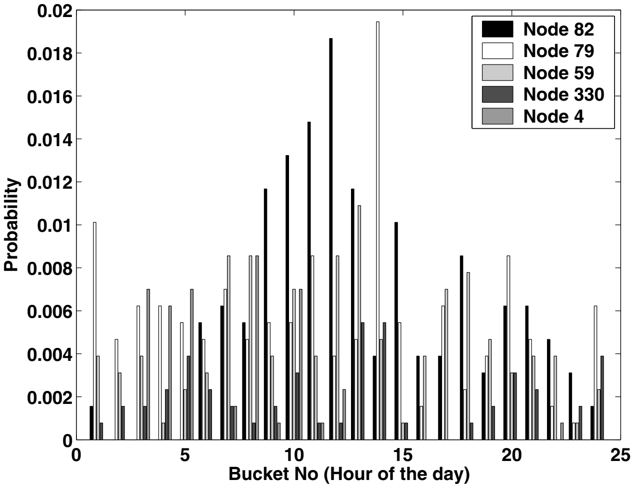


Figure 6. Distribution of daily failures.

those found at node 59. The failures encountered at node 4 tend to be more clustered, where essentially no failures occur in hours 1 and 2, hours 13 through 21, and hours 23 and 24, and the largest probability mass of failures occurs during hours 3 through 5, and hours 8 and 10.

The next issue of interest concerns how these temporal correlations vary over longer time horizons. In Figure 7 we plot the number of raw failures per hour at the 5 nodes with the most failures over the entire duration of our time-series dataset. These results clearly illustrate that the temporal correlations vary over time and that the patterns of such behavior differ among the individual nodes. In particular, Node 82 experiences no failures for the first quarter of the time-series and then it encounters bursts of failures that are fairly well spread out over the remainder of the time-series, with the majority of bursts being of size 3, a few being of size 4, and a single burst of size 5. No failures are experienced at node 79 for almost the entire second half of the time-series, while the first half contains bursts of failures that are more clustered than those at node 82, most of which are of size 3. With few exceptions, the failures at node 79 dominate the system failures during the first half of the time-series. Node 59 encounters no failures for more than the first third of the time-series and then it experiences bursts of failures that are fairly well spread out over the remainder of the time-series, with the majority of bursts being of size 1, several being of size 2, a few being of size 3, a couple being of size 4, and a single burst of size 5. No failures are experienced at node 330 for almost the entire first half of the time-series, followed by a cluster of failure bursts of sizes 1–3 close to the middle of the time-series. Node 330 also encounters another cluster of somewhat similar failure bursts towards the end of the time-series, with several single failures spread out in between the two clusters of failures. The failures experienced at node 4 are quite bursty and dispersed, with all of these failures concentrated at 16 points in time over the entire duration of our time-series dataset. Node 4 also encounters the maximum hourly burst of 9 failures among the top 5 nodes.

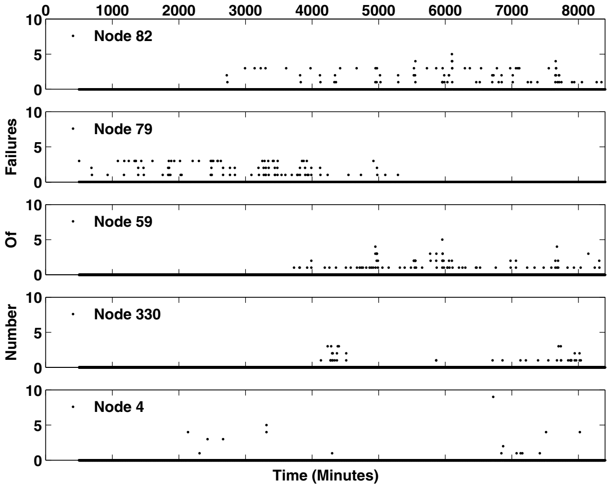


Figure 7. Failures as a function of time.

We feel that the failures for these nodes, and their time-varying behavior, are influenced considerably by the load that is imposed on them. Investigating such correlations between the nature and intensity of the workload, and the failure properties is part of our future work.

5. Concluding Remarks

Evolving technological trends and growing system complexity necessitate taking failures into account when designing systems for the next generation. This issue becomes particularly important when designing and deploying servers that must cater to the needs of hundreds of users without significant disruption of service. While there are several (proactive and post-active) techniques for dealing with and mitigating the impact of failures, there is a critical need to understand the failure behavior of real systems. Such an understanding can not only help evaluate and tune existing techniques, but also develop new mechanisms.

Towards this goal, we collected event logs of system health from 395 server class systems in a machine room at the IBM Thomas J. Watson Research Center over a year. This production environment contains several heterogeneous servers – from multiprocessor (2/4/8/12-way) SMPs to clusters – running a diverse spectrum of workloads – from traditional file and email services, to commercial applications such as transaction processing and even long running scientific applications.

An examination of these logs demonstrates that although improvements in system robustness continue to limit the number of actual failures to a very small fraction of the recorded errors, the failure rates are still significant and highly variable. The results of our analysis also show that the underlying stochastic error and failure processes exhibit time varying behavior and different forms of strong correlation structures, including possible long-range dependence, significant periodic behavior, and considerable cross-correlation between less-serious errors and more-serious errors plus failures. In particular, the system error and failure patterns are clearly nonstationary and they consist of relatively long time intervals that are stationary, many spanning more than a day. Moreover, the failures are not uniformly distributed among the nodes and a small fraction of the nodes incur most of the failures, with less than 4% of the nodes experiencing almost 70% of the failures. The failures at the nodes incurring the most failures also have a strong temporal correlation with time of day at the hourly level and these temporal correlation patterns vary over time with such behavior differing among the individual nodes.

There are several interesting directions for future work, in terms of using the data collected as well as further work in data collection itself. Using this data, we would like to evaluate previously proposed techniques for enhancing system availability. In addition, we would like to explore the possibility of predicting failures based on events occurring at a node, or across the network, and their usefulness in proactive system maintenance. In terms of data collection, we would like to procure more data not only from other environments, but even within the production system studied here in the hope of gaining more insights on how hardware and software advancements, and workload evolutions, impact the failure properties.

Acknowledgements: We thank H. Ombao and B. Ray for making available to us an implementation of the Auto-SLEX procedure. We also thank Dr. R. Iyer and the anonymous referees for invaluable comments which helped us improve the quality of the presentation. A. Sivasubramaniam is supported in part by NSF grants 0325056, 9988164, 0097998 and 0130143, an IBM faculty award and an IBM SUR equipment grant.

References

[1] Power and Temperature-aware Computing. Special Issue, *IEEE Computer*, 36(12), 2003.

[2] X. Castillo and D. P. Siewiorek. A workload dependent software reliability prediction model. In *Proc. Intl. Symp. Fault-Tolerant Computing*, pp. 279–286, 1982.

[3] T. Heath, R. P. Martin, and T. D. Nguyen. Improving cluster availability using workstation validation. In *Proc. ACM SIGMETRICS Conf. Measurement and Modeling of Computer Systems*, pp. 217–227, 2002.

[4] IBM Corporation. Problem solving and troubleshooting in AIX 5.l. *IBM Redbooks*, 2002.

[5] R. Iyer, D. Rossetti, M. Hsueh. Measurement and modeling of computer reliability as affected by system activity. *ACM Trans. Computer Systems*, 4(3):214–237, 1986.

[6] R. K. Iyer and D. J. Rossetti. Effect of system workload on operating system reliability: A study on IBM 3081. *IEEE Trans. Soft. Eng.*, SE-11(12):1438–1448, 1985.

[7] S. Karlin and H. M. Taylor. *A First Course in Stochastic Processes*. Academic Press, Second edition, 1975.

[8] I. Lee and R. K. Iyer. Software dependability in the tandem guardian operating system. *IEEE Trans. Soft. Eng.*, 39(4):455–467, 1995.

[9] M. Lyu and V. Mendiratta. Software fault tolerance in a clustered architecture: Techniques and reliability modeling. In *Proc. 1999 IEEE Aerospace Conf.*, pp. 141–150, 1999.

[10] J. Meyer and L. Wei. Analysis of workload influence on dependability. In *Proc. Intl. Symp. Fault-Tolerant Computing*, pp. 84–89, 1988.

[11] S. Mourad and D. Andrews. On the reliability of the IBM MVS/XA operating system. *IEEE Trans. Soft. Eng.*, 13(10):1135–1139, 1987.

[12] H. Ombao, J. Raz, R. vonSachs, and B. Malow. Automatic statistical analysis of bivariate non-stationary time-series. In *J. American Statistical Association*, Vol. 96, pp. 543–560, 2001.

[13] A. Radovanović, B. Ray, and M. S. Squillante. Statistical properties of traffic patterns in large-scale commercial Web sites and their performance implications. Technical report, IBM Research Division, 2003.

[14] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on soft error rate of combinational logic. In *Proc. Intl. Conf. Dependable Systems and Networks*, pp. 389–398, 2002.

[15] M. Sullivan and R. Chillarege. Software defects and their impact on system availability – A study of field failures in operating systems. In *Proc. Int. Symp. Fault Tolerant Computer Systems*, pp. 2–9, 1991.

[16] D. Tang, R. K. Iyer, and S. S. Subramani. Failure analysis and modelling of a VAXcluster system. In *Proc. Intl. Symp. Fault-tolerant Computing*, pp. 244–251, 1990.

[17] D. Tang, R. K. Iyer. Impact of correlated failures on dependability in a VAXcluster system. In *IFIP Working Conf. Dependable Computing for Critical Applications*, 1991.

[18] K. Vaidyanathan, R. E. Harper, S. W. Hunter, and K. S. Trivedi. Analysis and implementation of software rejuvenation in cluster systems. In *Proc. ACM SIGMETRICS Conf. Measurement and Modeling of Computer Systems*, pp. 62–71, 2001.

[19] J. Xu, Z. Kallbarczyk, and R. K. Iyer. Networked windows NT system field failure data analysis. *Technical Report CRHC 9808 University of Illinois at Urbana- Champaign*, 1999.

[20] J. Zeigler. Terrestrial Cosmic Rays. *IBM J. Research and Development*, 40(1):19–39, 1996.