Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 08-014

Failure Classification and Inference in Large-Scale Systems: A Systematic Study of Failures in PlanetLab

Sourabh Jain, Rohini Prinja, Abhishek Chandra, and Zhi-li Zhang

April 24, 2008

Failure Classification and Inference in Large-Scale Systems: A Systematic Study of Failures in PlanetLab

Sourabh Jain, Rohini Prinja, Abhishek Chandra, Zhi-Li Zhang
Department of Computer Science, University of Minnesota - Twin Cities
{sourj, rohinip, chandra, zhzhang}@cs.umn.edu

Abstract

Large-scale distributed systems are prone to frequent failures, which could be caused by a variety of factors related to network, hardware, and software problems. Any downtime due to failures, whatever the cause, can lead to large disruptions and huge losses. Identifying the location and cause of a failure is critical for the reliability and availability of such systems. However, identifying the actual cause of failures in such systems is a challenging task due to their large scale and variety of failure causes.

In this work, we try to understand failures in a large-scale system through a two-step methodology: (i) classifying failures based on their statistical properties, and (ii) using additional monitoring data to explain these failures. We illustrate our methodology through a systematic study of failures in PlanetLab over a 3-month period. Our results show that most of the failures that required restarting a node were of small size and lasted for long durations. We also found that incorporating geographic information into our analysis enabled us to find site-wise correlated failures. We were also able to explain some failures by using error-message information collected by the monitoring nodes, and some of short-lived failures by transient CPU overloads on machines.

1 Introduction

Large-scale distributed systems such as content distribution networks [1], peer-to-peer systems [3, 4], computation Grids [7, 6, 10, 11], and network testbeds [8] provide an essential platform for numerous distributed applications ranging from content sharing and Web services to VoIP and scientific simulations. Many of these systems consist of large number of nodes communicating with each other over widely distributed networks. Due to their inherent scale, diversity and complexity, these systems are prone to frequent failures, which could be caused by a variety of factors: network instabilities, power outages, node crashes, application software failures or security attacks. Any downtime due to failures, whatever the cause, can lead to large disruptions and huge losses; a recent example of such a failure is the one affecting Skype users in August 2007 in which 200M users could not use the Skype service for nearly 48 hours [5].

To provide high level of reliability and availability in such large-scale systems, it is critical to detect, diagnose, and fix these failures as they happen. Identifying the location and cause of a failure is especially important for such troubleshooting. Accurate diagnosis can not only lead to quick repair and recovery of failures, but also provides insights for making longer-term decisions on resource provisioning, software debugging, and hardware upgrades. For instance, the ability to determine that a failure occurred at an end-host rather than in the network can help system administrators focus their troubleshooting effort on that end-host. Moreover, they can come up with the correct remedy if they can determine whether the failure was due to a hardware crash or a software bug. However,

identifying the actual cause of failures in large scale systems is a challenging task due to several reasons. First of all, simply identifying the failure events themselves does not provide enough information about their cause, or the possible remedy. For instance, discovering that a node has stopped responding does not indicate whether it has suffered a node failure, a link failure, or a software crash. The problem is further aggravated by the presence of heterogeneous computing environment consisting of variety of hardware, software and network conditions. This necessitates the need of a systematic approach to understand failures. In this work, we endeavor to develop a methodology to better understand failure patterns in a large-scale system. As part of this methodology, we employ additional information, such as data collected by monitoring systems and system logs that are routinely collected in such systems.

We use a two-step methodology to identify failures and their possible causes: (i) classifying failures based on their statistical properties, and (ii) using additional monitoring data to explain these failures. We illustrate our methodology through a systematic study of failures in PlanetLab over a 3-month period. Besides providing interesting insights into the behavior of PlanetLab, an important goal of this study is also to identify the strengths as well as limitations of our methodology and the requirements on monitoring data that would make it suitable for failure analysis.

We use 3-month data collected by CoMon [16], a centralized monitoring system running on PlanetLab, for our analysis. CoMon collects node and slice-level (application-level) information of all PlanetLab nodes over time. First, we categorize the failures in meaningful classes according to their duration, group-size (number of nodes failing together), and nature of failure (hard–machine failures, or soft–network/software failures). Our results show that most of the hard failures are small-size and last for medium or long durations. On the other hand, the class of failures which were large-size and small-duration are mostly soft in nature. Secondly, we use other data such as location of a node, application and machine-level resource usage, etc. to make inference about failure causes. For instance, our results indicate that incorporating geographical location of nodes into our analysis helped us find site-wise failure correlation, providing us with better insights than those based on failure group-size alone, as has been done in previous studies [14].

The rest of the paper is organized as follows. Section 2 provides description of the dataset followed by an overview of our data analysis methodology. Sections 3 and 4 present details on classification of failures and inference using additional data respectively. Section 5 discusses related work followed by concluding remarks in Section 6.

2 Data Description and Methodology

2.1 Dataset Description

PlanetLab deploys a centralized monitoring infrastructure called CoMon [16] that collects and reports statistics on active PlanetLab nodes. The main purpose of this system is to enable PlanetLab administrators to spot problematic machines and/or slices running on them. The system is designed in such a way that a central monitoring node located in Princeton queries daemons running on the remote nodes every five minutes and collects data. The data consists of various metrics such as uptime, CPU and memory utilization, CPU load etc. Some of these metrics are actively measured by CoMon, e.g., CPU available to a spin-loop program (i.e. Free CPU), while other metrics are passively measured or synthesized, e.g., number of live slices. If a PlanetLab node is reachable, it responds to the queries with its node-level/slice-level information, which is written to a central repository. If a node is unreachable, the central monitor simply records a query failure with a corresponding error message. The archived data resides in the repository which is publicly accessible through the CoMon website [2]. For our analysis, we used a 3-month long (Dec'06-Feb'07) data trace collected by CoMon.

CoMon collects a large amount of monitoring data from the PlanetLab nodes and archives it in textual format to a set of files, each corresponding to a day of collected data. We preprocessed these archive files to get the data

in a format suitable for our analysis and to extract important metrics from a large set of metrics available. This was done by reducing the data to a time-series representation for each of the interesting metrics which helped us visualize the data better. To extract failure characteristics, we organized the data such that the failure time series data for each node (i.e., the series of 0's and 1's corresponding to a node being responsive or not) was matched to the corresponding time series of various other metrics returned by CoMon (such as CPU, memory, and bandwidth usage). Having the data in this format simplified the statistical analysis of the failures as well as their possible causes.

2.2 Data Analysis Methodology

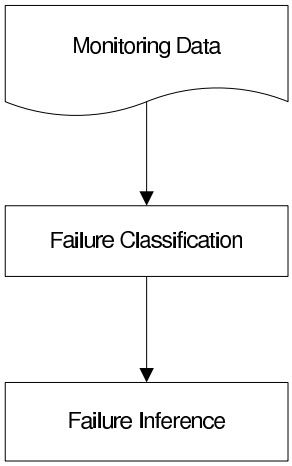

Figure 1. Data analysis methodology.

We followed a two-step methodology to identify failures and their possible causes (Figure 1). At a high level, this methodology consisted of first separating out failures based on their statistical properties, and then using other monitoring data to explain these failures. The first step consisted of classifying the various failures observed in the data based on their characteristics such as their duration (how long did a failure last), size (how many nodes failed together), and whether a failure was hard (node failure) or soft (software/network failure). This classification was largely done based on the failure time series data itself, and the goal was to separate different kinds of failures, since failures with different characteristics are likely to have different causes. The second step consisted of failure inference, where we correlated the failures in each class to additional monitoring data, such as location of node, resource usage, and types of error messages. The goal of this step was to be able to explain the causes of the various failure classes based on additional information available. Next, we present results obtained from our analysis using this two-step methodology.

3 Failure Classification

The goal of failure classification is to separate failures into classes, with each class of failures having common symptoms and/or cause. In order to arrive at an appropriate classification, we need to know the important characteristics/attributes of failures. We begin by looking at the overall distribution of node-availability with respect to time. This distribution is shown in Figure 2; x-axis represents time (from Dec 1, 2006 to Feb 28, 2007) and y-axis represents nodes sorted on their geographical coordinates. A horizontal row in the figure represents availability

pattern of a node over the chosen time interval[1].

We see that almost all nodes fail over time with some nodes failing more often than others. Figure 2 also highlights varying characteristics of node-failures. Some failures last for a very long duration while others are very short; some failures affect all the nodes while others affect only a small group of nodes. We also see instances of correlated failures: failures on nodes present in different geographical locations occurring at the same time and for the same duration (marked as 1), failures on nodes present in the same location (marked as 2) and failures in which almost all nodes in the system fail at the same time(marked as 3).

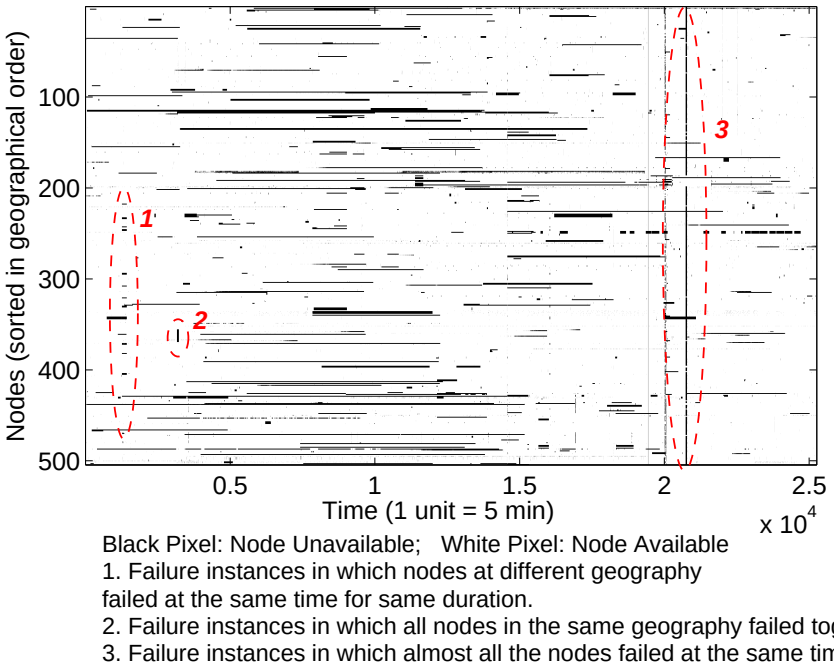

Figure 2. Availability time-series of nodes in PlanetLab for 3-month period.

The key observation in this figure is that failures show many diverse characteristics in a large-scale system. All the failures are not same, they differ in terms of their durations and sizes; these differences might be because of difference in the cause of failures.

Since the number of failures is large, we divide failures into different classes to make the analysis tractable. Classification is required and very useful as techniques to explain one kind of failure might not be applicable to analysis of other failures and looking at all the failures together might skew the results. At this point, we ask the following questions:

- How do we categorize failures? What attributes can we use to distinguish between failures?
- Can the failures in classes thus formed be explained with the given data?
- Which nodes are prone to what kind of failures?

We start by using two important attributes available from the node-availability data: failure duration and failure group-size. In the following subsections, we discuss their overall distribution.

[1]In Figure 2 we only show those failures which are contained entirely in the 3-month interval, i.e, we exclude the failures which occurred before Dec 1, 2006 or continued after Feb 28, 2006.

Table 1. Comparison of distributions for short, medium and long duration failures

Failure Type	Failure Frequency
Short Duration	15834
Medium Duration	5048
Long Duration	411

3.1 Failure-duration Classification

Failure duration is an important characteristic of a failure. Intuitively a failure that lasts for few minutes would have different cause behind it than a failure which lasts for more than a day. For instance, a transient overload or a network failure can cause a node not to respond for small duration but this may not last for a long time. A relatively longer duration failure can be caused by power outage, maintenance routines, etc., and still longer duration failures might happen due to hardware malfunction which requires human intervention and can take days to fix. To determine classes of failures based on duration, we look at the overall distribution of failure duration (shown in figure 3).

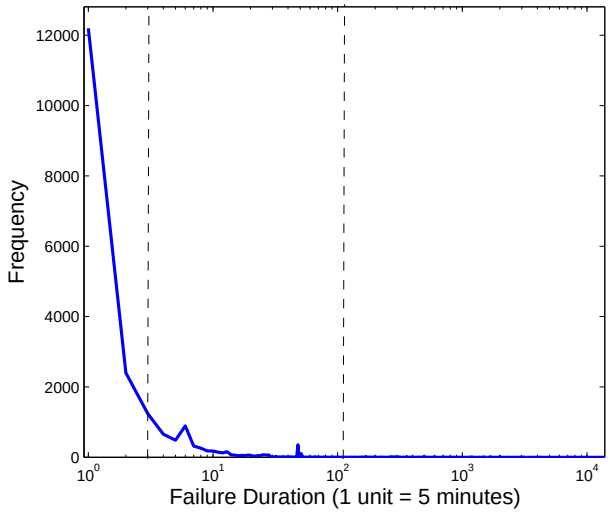

Figure 3. Distribution of failure duration.

Figure 3 shows that extremely short-duration failures are very high in number[2] with relatively fewer longer duration failures. For our discussion, we call a failure *short-duration* if it lasts for less than 15 minutes, *medium-duration* if it lasts for more than 15 minutes but less than 12 hours, and *long-duration* if it lasts for more than 12 hours. Table 1 shows the frequency of each of these failures. As seen from the Table, nearly 75% of the failures are short-duration.

3.2 Failure Group-size Classification

Having looked at classes of failures with respect to duration, we turn to investigate failures with respect to their group-size. Failures at different nodes that start at the same time may be caused due to the same event in the

[2]In figure 3 we see two spikes, one for failure duration of 6 units (30 minutes) and the second for failure duration 26-28 units (about 2.5 hrs); these spike corresponds to the global failures seen at monitoring node (marked as 3 in figure 2).

system. Figure 4 shows failure size distribution; most of the failures affect a small number of nodes only. In terms of numbers more than 90% of failures are affecting less than 5 nodes. Based on our conceptual understanding, we divide failures into two classes: we call a failure small-size failure if it affects less than 5 nodes else we call it a large-size failure. Any failure which involves almost all the nodes is referred to as a global failure; these failures are particularly interesting and important because such large-scale failures make the whole system unusable.

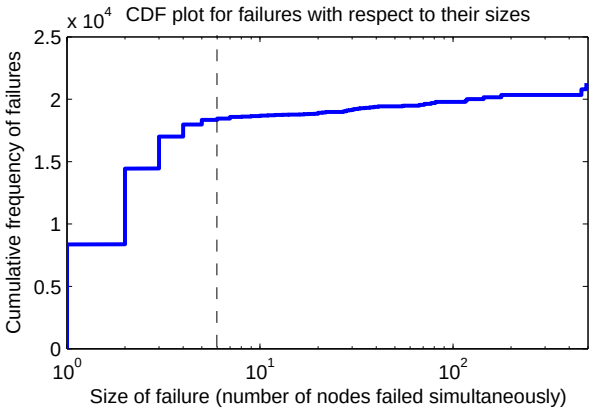

Figure 4. Distribution of failure size.

Global Failures: As shown in figure 2 there are 3 instances of global failures (marked as 3) in the 3-month data. Upon investigation, we found that in one of these instances, none of the nodes in PlanetLab appeared to be responding; a possible reason could be that monitoring node was down and did not record information for any of the nodes. In the second global failure instance, all of the nodes present in Europe and Asia were down[3]. In the last global failure instance, only a handful of Princeton nodes were up and all other nodes were down. Communicating with PlanetLab administrators, we came to know that during the same time Princeton site was going through a network upgrade. This explained why the monitoring node was unable to get data from other nodes, giving the illusion of their having failed. We do not include these failures in our further analysis.

So far we have used two attributes to define a failure: duration and group-size. However, these two attributes represent the symptoms of failures; they are not sufficient to help us understand what caused the failures in the first place. Moreover, these attributes by themselves may provide incomplete information. For instance, as seen for the case of global failures, these failures actually were not node failures, but possibly a network failure or a monitoring side problem. In particular, another question we would like to answer is whether a failure is a machine level failure, or is it caused by a network failure or a software bug.

3.3 Hard and Soft Failure Classification

In previous studies [9, 13, 14] on failures in large-scale systems, a node is assumed to have failed if it cannot be reached from other nodes. A node can be unreachable due to two possible reasons: (i) the node is actually down (a hard failure) and (ii) the node is up but not responding (a soft failure). A soft failure might be caused by a temporary CPU overload at the node or it might arise due to a transient network congestion. On the other hand, a hard failure might be due to a software crash or a hardware failure which requires manual intervention to get the node back online.

To distinguish between hard and soft failures, we use the 'Uptime' information from CoMon trace. Every five minutes, CoMon records uptime as reported by the system uptime counter at every node; we detect hard failures as a reset of the uptime counter.

[3]For both these instances, we were not able to provide evidence for possible cause of the failure using the given resources.

Table 2. Comparison of distributions for hard and soft failures

	Hard Failures	Soft Failures
Frequency	623	20670
Total Downtime	1500 days	766 days
Mean Duration	34 hrs	53 min
Failures per node	Mean = 2.1	Mean = 41

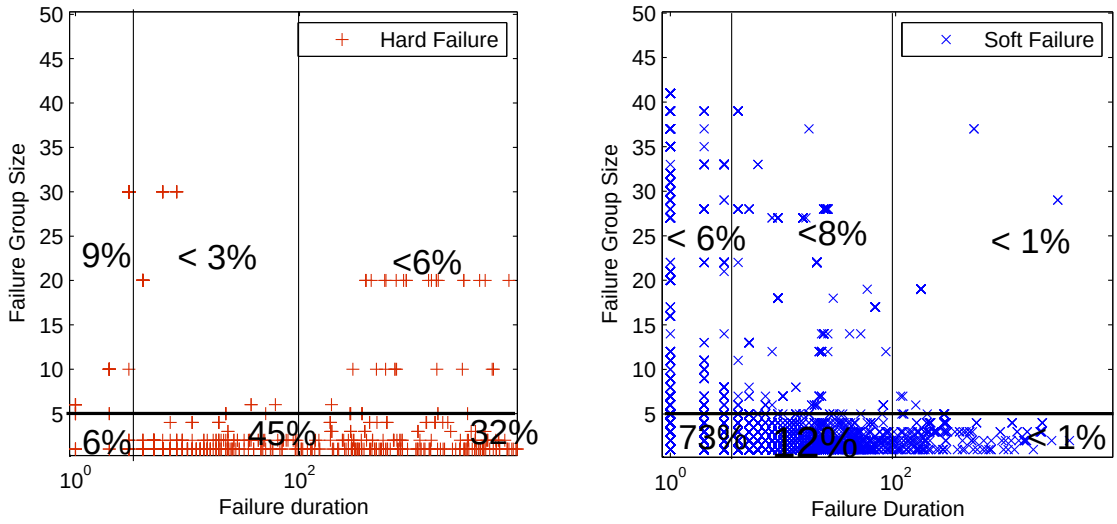

Figure 5. Hard and Soft failures in comparison to duration and size of failures.

Table 2 shows the distribution of hard and soft failures; most of the failures recorded in CoMon traces were soft failures accounting for more than 95% of total failures. Also, mean frequency of hard failures per node was only 2 compare to 41 for soft failures. However, the total downtime caused by soft failures was nearly half as compared to that of hard failures. Thus, although hard failures are less likely to occur, they can cause more downtime than the soft failures. These results show that it is important to consider both hard and soft failures to increase the overall availability of the system, but the remedy for each type may be different.

3.4 Relations between Failure Classes

So far, we have described our failure classification using three attributes: failure duration (short, medium, long), failure group-size (small, large, global) and failure type (hard or soft). Now we look at the relations between these failure classes, to see if the attributes are correlated with each other. In particular, we want to see if the type of failure could be explained based on its size or duration, or vice versa. Table 3 shows the number of hard and soft

Table 3. Overall distribution of hard and soft failures in different failure classes

	Short Duration		Medium Duration		Long Duration	
	hard	soft	hard	soft	hard	soft
Small Size	21	14570	308	3095	192	158
Large Size	16	413	25	137	31	19

failures occurring in each combination of duration- and size-wise class. Figure 5 shows the same data, except that it presents the number of hard and soft failures in each class a percentage of the total hard and soft failures, respectively. We make some interesting observations from these results:

1. Most of the hard failures are small size and medium or long duration failures (see Figure 5). We conjecture that hard failures may be caused due to manual shutdown, hardware failure, power outage, etc., because of which they do not affect a lot of nodes simultaneously. Furthermore, human intervention is required to get the nodes back up after shutdown, thus they are of longer duration.

2. Most of the large size failures of short and medium duration are soft failures (see Table 3), i.e., the nodes are actually up though they fail to respond to the monitoring node. This could be either due to large-scale network failures in the Internet or due to failures at the monitoring node such that monitoring node failed to get responses from other nodes. Such failures are unlikely to last long, as network traffic would typically get rerouted around failures soon, and monitoring node-side failures are also likely to be fixed by PlanetLab administrators in a timely fashion.

3. Most of the small-size, short-duration failures are soft failures (see Table 3), and vice versa (see Figure 5). We conjecture that this could be due to transient overloads on nodes, which could result in short periods of non-responses. Such overloads are likely to occur randomly and last for short durations.

The classification of failures and their observed correlations provide us with some insights and conjectures on the possible behavior and causes of failures. They also provide us with a useful filter to distinguish between different kinds of failures, so that we can study unrelated failures separately. In the next section, we use additional monitoring data to refine our understanding of failure causes, and explain them based on the available information.

4 Failure Inference

In this section, we refine the insights gained through failure classification by using additional information available from the monitoring data to explain some of the observed failures. In particular, to test some of the conjectures formulated based on the observed characteristics of the various failure classes, we use three sets of information available from the CoMon data for failure inference: (i) Geographical location of nodes, (ii) Error messages seen by the monitoring node during the failure, and (iii) Node level resource usages. In the following subsections, we provide the results of failure inference using these additional metrics, and show how it helps us in explaining subsets of failures.

4.1 Geographic Location

Nodes in PlanetLab are distributed at different sites located at various locations in the world and geographical location of each node is captured using the longitude/latitude of it's site. We use this information to detect site-wise correlated failures; a failure is site-wise correlated if all (and only) the nodes from the same site fail together.

To find possible geographical correlation among node failures, we consider only small size failures with sizes 2-5, since a large number of sites (65% of the total sites) contribute only two nodes to the PlanetLab system. Table 4 shows the median value of pairwise distances for failed nodes for each failure size from 2-5. We see that there were a large number of failures which occurred together on machines that were co-located. In particular, we see there were 593 failures of size 2, indicating a site-wise failure of 2 machines at the same site. These results show that lot of small size failures can be attributed to geographical events which makes all the nodes at the same site unavailable from the monitoring node.

Overall, we found that 6.41% of all the failures in the 3-month duration are site-failures, of which 7.73% are hard and 92.27% are soft. We also found that most of the soft failures that were site-wise correlated were

Table 4. Distribution of small size failures with respect to geographical distances.

	Median value of pairwise distances for failed nodes		
Failure Size	0 km	1-200km	$>$ 200km
2	593	186	3223
3	69	24	1353
4	17	13	433
5	13	13	118

either short-duration or medium-duration whereas fewer site-correlated hard-failures were of short duration. These results indicate that most of the soft failures at different sites had been handled within 12 hours as opposed to hard failures which had longer durations.

The key take-away message from these results is that even small size failures could be correlated, as we see from the PlanetLab data, which were mainly due to geographical events affecting nodes at a given site, and large-size failures might be uncorrelated occurrences, at least in terms of proximity of occurrence. Next we explore another possible indicator of failure types, namely, error messages seen during the failures.

4.2 Error Messages

In CoMon, the monitoring node writes an error message to the trace when a remote node cannot be contacted. The combination of error messages seen during a failure can provide some insights into the cause of the failure. For instance, if a "No Response" error message is followed by a "Connection Refused" error message, it may imply that node was unreachable in the beginning and then the monitoring daemon was not running properly. With each failure in our data-set, we associate the sequence of error messages seen during that failure. Before we go into the results, we first explain the different types of error messages seen by the monitoring node in PlanetLab.

- Connection Refused (CR): This error message usually results from trying to connect to a service that is inactive on the remote host. In our context, it means that although the node was up, the monitoring daemon was not running on that node.
- Connection Reset by Peer (CRP): This error message results from the loss of connection on the remote socket due to a timeout or a reboot.
- No Route to Host (NRH): This error implies either a network problem near the remote node, or that node is not up.
- Network is Unreachable (NU): This error implies a network problem near the monitoring host when it cannot reach any of the nodes.
- No Response (NR): It is not a standard socket error message, therefore does not provide any additional information.

In Figure 6, we plot the probability of seeing different sequences of error messages for hard and soft failures. We found that most of the failures involving sequence of error messages "No Response" and "Connection Refused" were hard failures. Furthermore, a failure involving a sequence of "No Response", "Connection Refused" and "No Route to Host" was most likely to be a hard failure. We also see that during most of the short-duration soft failures, the monitoring node did not receive any error messages for its queries. It could happen that the node had some network problem, but intermediate routers had suppressed the ICMP error messages. In our analysis we

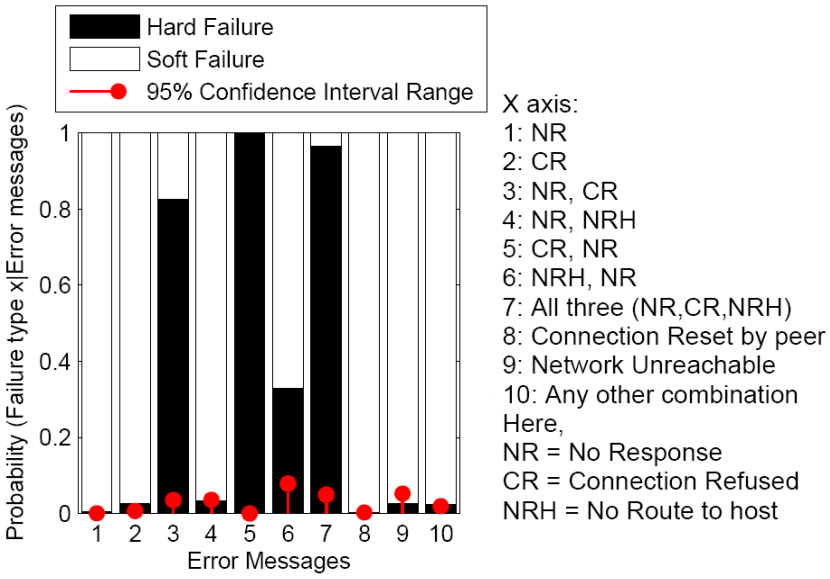

Figure 6. Probabilities of seeing an error message for hard/soft type of failures

also found that failures as a result of Internet failures were resolved relatively quickly. Very few of the failures which saw 'Network Unreachable' error message were of long-duration. Moreover all the failures involving error messages such as 'Network unreachable' or 'Connection reset by peer' imply a network failure and hence these were found to be instances of soft failures; we found that more than 95% of such failures were soft.

We analyzed the data to see if error message combinations were able to distinguish between different classes of failures that we see in Figure 5. We noticed that hard failures have mostly NR—CR combination and soft failures have NR. Soft failures that are medium-duration and large-size have predominantly CR error message, which suggests the failure of monitoring daemon on the corresponding nodes.

Another interesting pattern in error messages appeared for some of the hard failures when we observed a repetitive sequence of 'Connection Refused' and 'No Response' error messages. As we know that 'Connection Refused' error message is sent by the socket layer at the destination host when it cannot find the process corresponding to the requested service (port). It implies that node was up, followed by one or more hard failures. Since no information is recorded locally, nothing can be said about these kind of failures.

In summary, here we presented the results of correlating different type of error messages to hard and soft failures. Our results showed that there are sets of error messages which can be used to distinguish between two classes of failures. e.g. a sequence of error messages "No response" and "Connection refused" is most likely to be seen during a hard failure. On the other hand an appearance of "No Response" and "Connection Refused" alone during a failure indicates an instance of soft failure. Though these error messages provide a good indication of possible failure type (in terms of hard/soft failure), still it did not explain many subclasses of failures such as short duration failures or small size geographically un-correlated failures etc. In order to explain the behavior of these subclasses of failures we tried to correlate them with node level attributes such as CPU and memory usage. Next we look at the analysis of these attributes.

4.3 Machine-Level Attributes

We analyzed the monitoring data to see if the machine-level metrics like CPU-usage, Free CPU or Memory Usage correlate with certain types of failures.

CPU Overloads: In order to investigate the correlation between failures and CPU overloads, we compared the distribution of free CPU on the nodes just before the failure with overall distribution of free CPU. These results are shown in the Figure 7. We found that as the value of Free CPU goes below 10%, chances of failure were increased by 3 times. Figure 7 plots the probability of having a failure if the value of Free CPU is in given range. It is interesting to see that $P(\text{Failure happens } |0\% \leq \text{FreeCPU} < 10\%)$ is 3 times higher than $P(\text{Failure happens } |10\% \leq \text{FreeCPU} < 20\%)$, and it decreases with the increasing Free CPU. These results show that failures can be correlated with CPU overloads. Also, in Figure 7 we plot the probability that Free CPU was in a certain range given that there was a failure just after it. As seen in the Figure $P(0\% \leq \text{FreeCPU} < 10\% \mid \text{Failure})$ is more or less 25 % compared to $P(0\% \leq \text{FreeCPU} < 10\%)$ which is less than 7.5%.

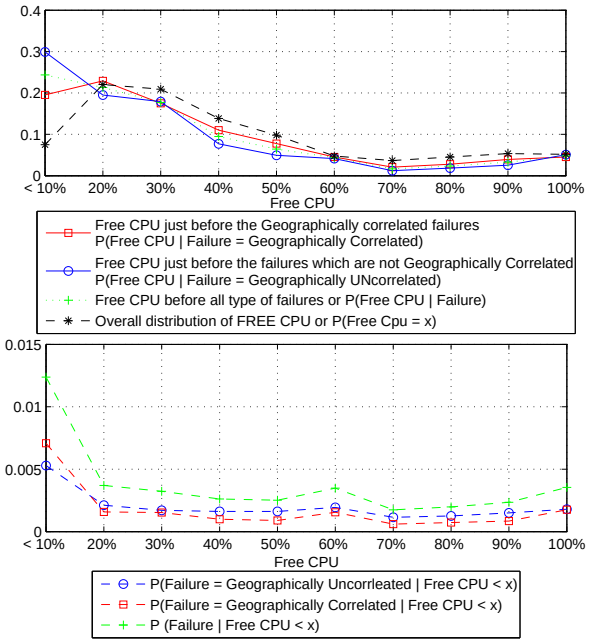

Figure 7. CPU overloads and failures.

Figure 7 showed that failures could be correlated with higher CPU usages. Next we focus on failures which are more likely to be caused by the transient CPU overloads. Therefore we hypothesized that CPU overloads are more likely to be responsible for short duration failures which do not occur in the same geography. We investigated our hypothesis by correlating short duration failures with CPU usage. We divided the set of short duration failures into two sets: one set consisting of failures that are geographically correlated and the other set consisting of failures that are not geographically correlated. The reason behind this division of failures was that geographically correlated failures are more likely to be caused by geographical events such as local network failure and therefore are unlikely to be correlated with factors such as transient CPU overloads. On the other hand, non-geographically correlated short duration failures are likely to be correlated with CPU usage. We show these results in Figure 7. In more than 30% of the correlated failures where multiple nodes were not from same physical location, very low FreeCPU was reported just before the failure; the percentage drops to 20% for geographically correlated failures. These results confirm our hypothesis that geographically independent short duration failures are likely to be correlated with transient CPU overloads.

Using our analysis, we found that short duration geographically independent failures are correlated with CPU overloads.

Memory Overload: Slices in PlanetLab run under a strict memory constraint [8]. In the environment provided by PlanetLab all the slices are restricted to use only a fixed amount of maximum memory. This constraint avoids the failures happening due to memory overloads. We verified our hypothesis by running similar experiments that we performed for CPU overloads, and no correlation was observed for the memory overloads and failures.

4.4 Section Summary

In this section, we showed that additional information provided in the traces can be used to explain certain kinds of failures. Our results show that some of the failures were geographically correlated. We also observed some interesting patterns for error messages seen during a failure. We showed correlation between free CPU available on a node and geographically independent short duration failures that it went through. These experiments provided us with some explanation on why certain failures happen. However, using the monitoring data available to us, we were unable to explain a large set of failures.

4.5 Contributions and Lessons Learned

We learned several useful lessons using our case study of failures in PlanetLab. Here is a short summary of the contributions of this study as well as the lessons learned.

- *Multi-dimensional Failure Classification:* In our study we classify the failures using failure size, failure duration and failure nature (hard/soft). We also showed typical characteristics of hard and soft failures with respect to size and duration. This classification helped us to distinguish between failures with different characteristics, and possibly different causes.

- *Failure Inference using additional information:* We explained some of the failure instances by using additional information available from CoMon. We used following attributes for the failure inferences. i) Node Location, ii) Error messages, iii) Node level resource usages (such as CPU/memory usage). We showed that geographical events are likely to cause small size correlated failures, Error messages helped us in distinguishing hard failures from soft failures, and higher CPU usage was found to be correlated with small size short-duration failures which were geographically independent. This failure inference enabled us to utilize additional monitoring information to explain different kinds of failures.

- *Limitations of failure inference:* CoMon uses a centralized monitor, which is sufficient for the scale of PlanetLab, and also serves the primary purpose for which CoMon is set up. We performed failure inference using the information available from the monitoring system. However, using this data to infer failures has the pitfall that monitor-side failures can be falsely attributed to system-wide failure events. This phenomenon is observed as the vertical black line in Figure 2, where a monitoring-side network failure prevented the central monitor from connecting to other nodes in PlanetLab. Moreover, for several failures, inference was limited to the monitoring information provided by CoMon, which could be explained better with access to system logs or application-level information.

5 Related Work

Prior work in this area can be divided into two parts i) Failure analysis and prediction and ii) Monitoring systems. In the following section we provide a brief overview of some of the related works.

Failure Analysis and Prediction: Prior studies have focused on failure characterization [14], availability prediction [13], and churn characterization [9] in large-scale systems. Most of these studies have used node up-down information to characterize failures and any other data provided by the traces is not utilized. Statistical cross-node

Table 5. Comparison of different monitoring systems

Monitoring System	Design	Active/Proactive/Reactive	Monitoring Task
CoMon	Centralized	Active Monitoring	Node level resource monitoring
CoTop	Centralized	Active Monitoring	Distributed Application (slice) level resource monitoring
PlanetSeer	Decentralized	Reactive/Passive Monitoring	Failures in Wide-area networks
Ganglia	Distributed Monitoring	Active Monitoring	Cluster/grids

correlations are considered in [14], however, it provides limited insights into the reasons behind these correlations. In [14], failures are considered to be correlated if a large number of nodes failed together, however we showed in our research that even small-size failures can be correlated. Overall, while such studies may provide useful prediction/characterization models, they do not provide a better understanding of the failure causes, or the correct remedy to overcome their impact. In [18] authors analyzed the failures caused by different applications and in [17] the use of past failures to provide an automated diagnosis of future failures was suggested. However these works were limited to identification and diagnosis of node-level failures only and did not look at the interaction of failures occurring in overall large-scale system.

Monitoring: Monitoring data can be useful for failure analysis, and several monitoring systems have been deployed for large-scale systems. These include i) CoMon [16]: centralized resource monitoring system for the Planet-Lab, ii) CoTop [15]: a centralized resource monitoring system for the distributed applications in the Planet-Lab, iii) PlanetSeer [19]: a monitoring system for failures in the communication network, and iv) Ganglia [12]: a monitoring system for grid/cluster systems used for high-performance computing.

Table 5 shows a comparison of these different monitoring systems. Most of these systems have been designed for specific purposes; while the monitoring information provided by these systems can be useful, they are not inherently designed to provide metrics suited to failure analysis. For instance, CoMon has a centralized architecture, which is suitable for its intended use, but is prone to failures of the monitoring node itself. Similarly, while Ganglia uses a scalable and distributed design, which is more robust, it is not suitable for large-scale systems because of its reliance on a static tree topology, prior knowledge of node identities, and limited data aggregation capabilities.

6 Conclusions

Understanding the characteristics of failures in a planetary scale system is an important problem. Information about the type of failures and their possible causes can help in better resource allocation and hence better quality of service for distributed applications. In our study of failures in PlanetLab, we saw many diverse characteristics of failures in terms of their duration, group-size and nature. We proposed a systematic way of understanding the characteristics of failures by first classifying the failures into different groups followed by using additional information to ascertain their possible cause. Although it is very hard to determine the "true-cause" of failures by using the information available to us, in some cases we could infer that it was caused by local network, Internet or software mis-configuration. In most of the cases we were limited to information provided by the monitoring system.

References

[1] Akamai: Content distribution network. `http://www.akamai.com`.

[2] Comon-a monitoring infrastructure for planetlab. Available at `http://comon.cs.princeton.edu`.

[3] Gnutella. `http://www.gnutelliums.com/`.

[4] napster. `http://www.napster.com/`.

[5] Restarts cited in skype failure. Available at `http://www.nytimes.com/2007/08/21/business/worldbusiness/21skype.html`.

[6] D. Anderson. BOINC: A System for Public-Resource Computing and Storage. *5th IEEE/ACM International Workshop on Grid Computing*, pages 365–372, 2004.

[7] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@ home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[8] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[9] P. Godfrey, S. Shenker, and I. Stoica. Minimizing churn in distributed systems. In *Proceedings of the ACM SIGCOMM Conference*, sep 2006.

[10] S. Larson, C. Snow, M. Shirts, and V. Pande. Folding@ home and genome@ home: Using distributed computing to tackle previously intractable problems in computational biology. *Computational Genomics*, 2002.

[11] M. Litzkow, M. Livny, and M. Mutka. Condor-a hunter of idle workstations. *Distributed Computing Systems, 1988., 8th International Conference on*, pages 104–111, 1988.

[12] M. Massie, B. Chun, and D. Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004.

[13] J. Mickens and B. Noble. Exploiting availability prediction in distributed systems. *Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation-Volume 3 table of contents*, pages 6–6, 2006.

[14] S. Nath, H. Yu, P. Gibbons, and S. Seshan. Subtleties in tolerating correlated failures in wide-area storage systems.

[15] V. Pai. CoTop: A Slice-Based Top for PlanetLab.

[16] K. Park and V. Pai. Comon: a mostly-scalable monitoring system for planetlab. *ACM SIGOPS Operating Systems Review*, 40(1):65–74, 2006.

[17] C. Verbowski, E. Kıcıman, A. Kumar, B. Daniels, S. Lu, J. Lee, Y. Wang, and R. Roussev. Flight Data Recorder: Monitoring Persistent-State Interactions to Improve Systems Management. *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, 2006.

[18] C. Yuan, N. Lao, J. Wen, J. Li, Z. Zhang, Y. Wang, and W. Ma. Automated known problem diagnosis with event traces. *Proceedings of the 2006 EuroSys conference*, pages 375–388, 2006.

[19] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: internet path failure monitoring and characterization in wide-area services. *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation-Volume 6 table of contents*, pages 12–12, 2004.