The Shape of Failure

Taliver Heath, Richard P. Martin, Thu D. Nguyen
{*taliver, rmartin, tdnguyen*}*@cs.rutgers.edu*
Department of Computer Science
Rutgers University, Piscataway, NJ 08854

1 Introduction

The construction of highly available Internet services is an inexact science at best. Complete and partial outages due to a myriad of failures are still common. Indeed, as Internet services become woven into our everyday life, these failures are becoming important enough to make front-page news.

Part of the difficulty in constructing highly available services is that a modern Internet service is mind-boggling in complexity. The application, database, operating system, and firmware contain thousands of interfaces and millions of lines of code. Worse, unlike many other highly available systems, the major components are created by different organizations with potentially different goals and reliability parameters; no single entity has authority over the design and implementation of the entire system. Thus, ensuring traditional fault-tolerant techniques are followed top-to-bottom is a difficult, if not impossible, organizational task.

Fortunately, the situation is not completely intractable. A first step toward building failure-resistant Internet services is to decompose the system into high-level components separated by well-defined interfaces. Next, the fault behavior of each component can be characterized in isolation. Then, armed with an understanding of component behaviors, the designer of a service can architect the system to tolerate many possible faults. Toward this end, we focus on quantifying the behavior of the workstation nodes comprising the core of many services.

Our quantification takes the form of exploring the question: *what is the statistical distribution of reboots for workstations?* The Time-To-reBoot (TTB) is an important metric for service designers because it aggregates the failure and repair rates of many system components including the operating system, the workstation hardware, and the operator, into a single number that can be associated to a component with well-defined interfaces. Although a broad metric, we argue that TTB is sufficiently powerful and accurate to guide important design decisions about how to architect highly available services.

To explore the above question, we obtained the *last* logs of two clusters operating in very different environments: a cluster of 20 workstations in an undergraduate laboratory and a cluster of 20 workstations housed in a machine room and accessible only remotely by faculty members and graduate students. Our analysis shows the TTB for *both* clusters is best modeled as a wiebull distribution with a shape parameter < 1. This has important implications. First, the distribution is not memory-less, suggesting models using exponentially distributed times may not adequately describe node failure behaviors.

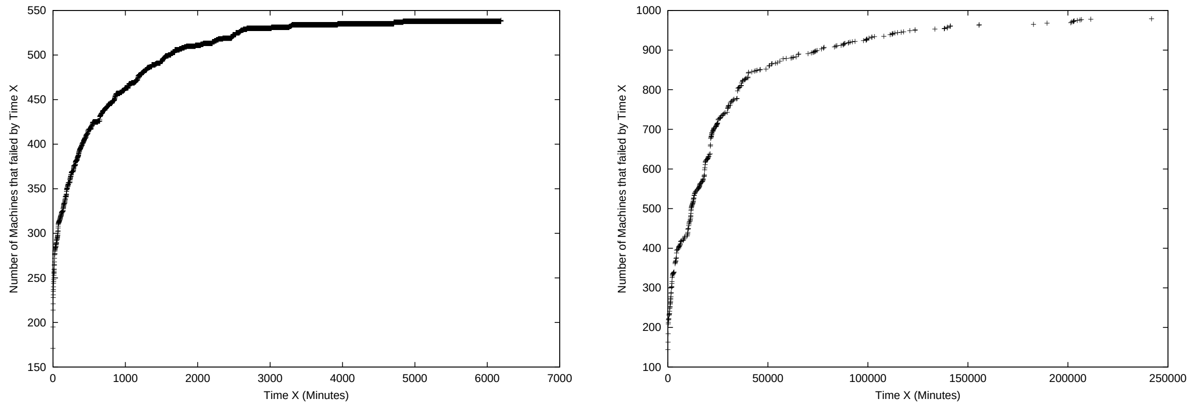
Second, a shape parameter < 1 implies that *workstations become more reliable with time.* That is, we can model a workstation as a component that becomes increasingly reliable the longer it has been operating. This has important ramifications for request distribution, data partitioning, and software rejuvenation.

In the remainder of the paper, we first describe how we obtained our reboot data and the method we used to perform the analysis. Next, in the results section, we discuss some implications of our observations. We then present some recent related work. Finally, we conclude with possible reasons why we observed the weibull distributions and offer avenues of future research.

2 Methodolgy

We gathered and analyzed data from two clusters. The first cluster consisted of 20 Sun Ultra-1 workstations running Solaris 5.8. All the machines were physically housed in one undergraduate laboratory and used exclusively for junior and senior computer science students. All machines were managed by a single system administrator.

The second cluster consisted of 20 machines that were physically housed in our machine room and were ac-



(a) (b)

Figure 1: **Number of failures as a function of time** *(a) Machine Room and (b) Undergrad Cluster failures plotted as if there were 521 and 979 machines, respectively, all started simultaneously, and times of failures recorded and plotted.*

cessible only via remote login to faculty and some graduate students. 17 of these machines were Sun Ultra-1 workstations and 3 were Sun Sparc-20 workstations. Unlike the first cluster, the supervision of these machines was spread across several different people that did not necessarily coordinate maintenance schedules.

We used the *last* logs to experimentally observe a machine's TTB history. The last logs stretched back between 6 months and 2 years. The undergraduate lab computers, which we call the *undergrad cluster*, recorded 979 reboots. The faculty computers, which we call the *machine cluster*, recorded 521 reboots.

The first step in our analysis is to compute the TTB from the restart information in the last logs. Note that there is a discrepancy between our observed TTB times and actual node uptimes, as the TTB includes down times. We were unable to remove this inaccuracy, however, because most machines did not record the crash/halt/shutdown time when they went down. However, we believe this inaccuracy would not impact our conclusions. We are currently investigating the relative size of difference between the uptime and TTB.

To examine the distribution of the data, we assume that the data is independent of reboots on other systems, or reboots on the same system. We will examine the validity of this assumption in the next section. We simulated an experiment with the cluster data by assuming that all computers were started at the same time, and then plotting the number of failures as time passes. This plot can be seen in Figure 1. Using these plots, we can determine the distribution of the data.

3 Results

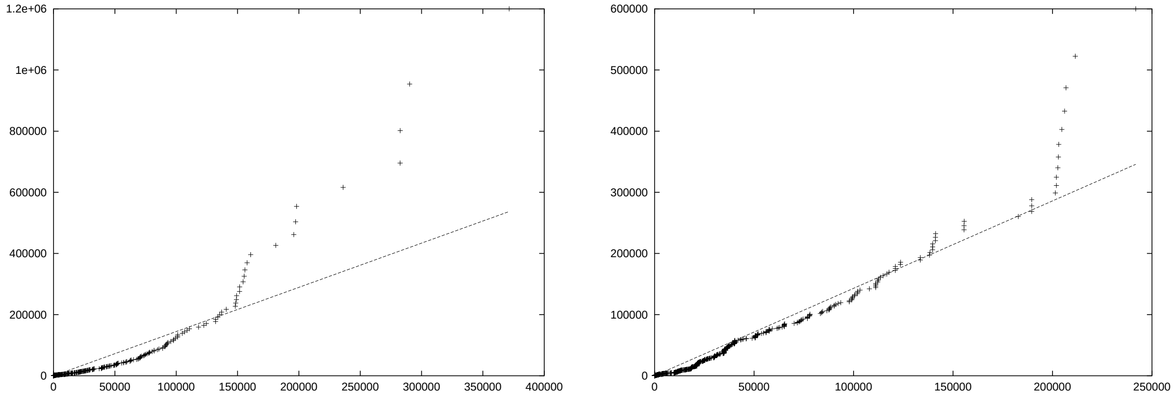
We attempted to fit our data to several distributions, including exponential, Weibull, Pareto, and Rayleigh. For each possible distribution, we used the Maximum Likelihood Estimates to approximate the parameters defining the "best" instance of the distribution for our data. After finding the parameters for a given distribution, we then used a quantile-quantile plot to compare the closeness of the distribution to our data (that is, how accurately this particular distribution models our observed data).

The quantile-quantile plot we used is described in [2]. This method arranges plots of known distributions against the given data. When the theoretical distribution matches the measured data, the quantile-quantile plot will give a straight line. Any variation away from the straight line indicates a deviation from the theoretical distribution. We then define the best fitting distribution as the one with the smallest residual from the least-square fitting of a straight line.

Applying the above method, we found that the Weibull distribution is the best matching distribution for data from both clusters. The quantile-quantile plots and the resulting fitted curves can be seen in Figure 2. The parameters for the best fit distribution is given in the following table:

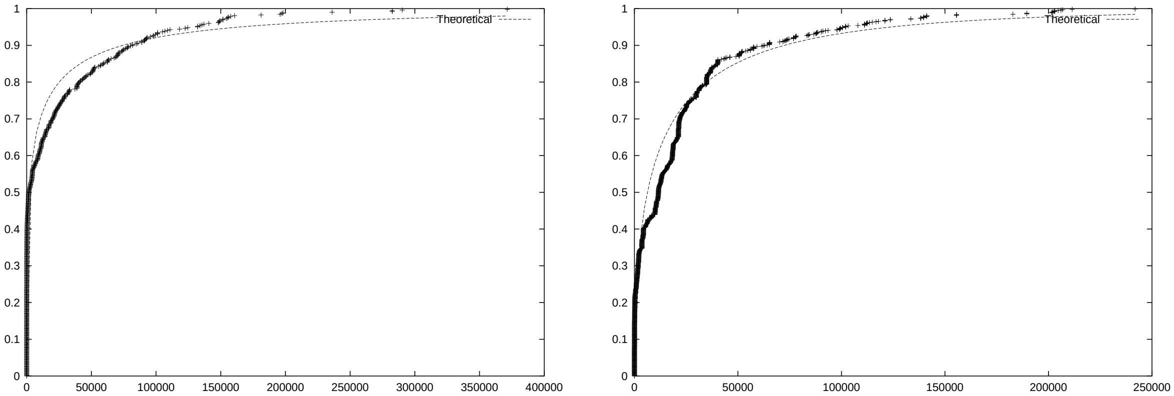
	Scale (a)	Shape (b)
Machine room cluster	99 hours	0.33
Undergrad cluster	219 hours	0.49

Since it fits the Weibull distribution with a shape parameter less than 1, we can see that the longer that a machine stays up, the less likely it will be rebooted



(a) (b)

Figure 2: **Fitting the Weibull Distribution to the TTB observations.** *Quantile-quantile plots showing the best fit curve for the (a) Machine Room Cluster and the (b) Undergrad Room Cluster.*



(a) (b)

Figure 3: **Fitting Weibull Distributions to the TTB observations.** *The resulting fitted curves are shown superimposed on the orginal data for (a) the machine room cluser and (b) the undergrad cluster.*

in the near future. The equation for the probability density function of the Weibull distribution is:

$$f(x) = 1 - e^{-\left(\frac{x}{a}\right)^b}$$

This analysis assumes that reboots between different systems are independent. It may be the case however, that events are related. For example, on the undergrad cluster the administrator would periodically upgrade system software, causing all of the systems to undergo reboots at approximately the same time. However, the machine room systems did not have this problem, so we believe that while these periodic reboots may have altered the distribution parameters slightly, they did not effect the general class of the distribution. An investigation as to correlation of reboots across machines in beyond the scope of this work.

4 Related Work

This short investigation does not include an exhaustive list of related work. Rather, in this section we present recent work most closely related to our own.

Perhaps the closest work related to this study is a recent work characterizing the behavior of of Microsoft Windows NT machines [3]. That study produced a detailed state-machine model of a workstation node, and thus requires fairly detailed knowledge of Windows NT. Another recent work examined the failure of components of an on-line image service [1]. That study focused more on the hardware components rather than the entire node.

5 Conclusion

When viewed as a complete system of hardware and software, we have found no indication that a system that has been "up" for an extended period of time is more likely to be rebooted than any other system. In fact, our results show the opposite: for Sun workstations running Solaris 5.8 in two of our departmental environments, a machine that has been up for a long period of time should be left up.

This initial result has many interesting implications. For example, the assumption behind many software and hardware rejuvenation techniques, that components decay over time, may be flawed, at least with respect to nodes viewed in their entirety (including hardware, operating system, and application software). It may be the case that, since our data encompass a myriad of event types that can lead to workstation "failures," the effect of component decay over time may have been overshadowed by other effects. Regardless, our results show that component decay does not seem to be the primary cause of workstation reboots so rejuvenation techniques are unlikely to be sufficient by themselves to make workstations more reliable.

A second implication is that the classic "bathtub" curve of high infant mortality, stable operating plateau, and finally increasingly failures may not apply to workstation clusters. However, our results did not cover time-scales on the order of many years. It may be the case that if we extended our observations to these time-scales, that we would see increasing failure rates. However, given that the useful operating range of computer hardware is about 3-years, and software even less, such long time regimes may be irrelevant to the construction of highly available services. Our results point to an avenue of future research that would quantify if such a bathtub curve exists for workstations in general, and what the actual shape is. We need to gather data on many different systems over a longer period to answer this question, however.

A third conclusion our initial results point to is that when a system starts, it may be far from a "clean" state. Indeed, configuration errors of different components may well cause the need for a subsequent reboots, leading to the idea that a machine's infant mortality period never fully goes away. This would also lend credibility as to why when a system has been up for a while, it tends to stay up. We have no evidence aside from our initial data of this claim being true, but it does warrant further investigation.

Finally, another interesting observation stems from the fact that although the undergraduate students had close, unsupervised physical contact with these systems, this fact did not change the shape or scale parameters of the observed TTB much. One would expect that machines where people could bump power cords and spill soda's on should have an different TTB distribution, but this was not the case.

Future research in this area could be aided by installing clusters with more detailed load recording information, since this was something that was not kept well and would have added much insight into other dimensions of failure analysis. Also, most Unix systems encourage periodic removal of the *last* logs. This may have been due to disk space concerns of the past, but these concerns have decreased with decreasing disk costs.

Unfortunately, in addition to inadequate logs, the practices of many IT shops thwart data collection in this area. For example, we have extensive reboot histories from another IT shop that includes MacIntosh, Windows 98 and Solaris machines. However, our analysis was limited by the policy of powering down the desktop machines every night. In addition, those systems were rebooted between users to reduce the chance of password spoofing. Even their Solaris servers were rebooted every week as a matter of course. In order to better understand the failure characteristics of machines, some machines must be left up as "controlled experiments". Of course, data from companies actually running cluster-based Internet services, as opposed to the more individual-oriented workloads that clusters in an academic environment typically support, would be invaluable.

References

[1] Asami, S. Reducing the cost of system administration of a disk storage system built from commodity components. Tech. Rep. CSD-00-1100, University of California, Berkeley, 2000.

[2] Jain, R. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, 1991.

[3] M. Kalyanakrishnam, Z. Kalbarczyk, R. I. Failure Data Analysis of LAN of Windows NT Based Computers. In *18th Symposium on Reliable and Distributed Systems, SRDS '99* (1999), pp. 178–187.