2011 30th IEEE International Symposium on Reliable Distributed Systems

A Characterization of Node Uptime Distributions in the PlanetLab Test Bed

Hakon Verespej and Joseph Pasquale
Department of Computer Science & Engineering
University of California, San Diego
La Jolla, CA, USA
e-mail: hakonv@microsoft.com, pasquale@cs.ucsd.edu

Abstract— **In this paper, we study nodes from the PlanetLab test bed to form a model of their uptime behavior. By applying clustering techniques to over a year's worth of availability data for the nodes, we identify six uptime distributions, each exhibiting unique characteristics shared by the nodes within it. The behavioral patterns exhibited by these groups, combined with the behaviors exhibited by the aggregate across the system, provide useful information for researchers designing applications that are run or tested on PlanetLab.**

Keywords-distributed system; classification; availability; modeling

I. Introduction

In distributed systems research, standard distributions such as Exponential or Weibull are often used to model node uptime [4][16]. These models may be appropriate depending on how and for what purpose they are used, but they do have limitations. In fact, it has been shown that standard distributions do not always accurately reflect actual system behavior [18].

In this paper, we study PlanetLab [15], a test bed consisting of hundreds of nodes hosted by a large number of institutions around the world and used for various research purposes including deploying and testing distributed applications. In observing the aggregate distribution of uptimes for PlanetLab, we find that it is not well-fitted to standard distributions due to some specific characteristics that are visible in the aggregate distribution.

Underneath the behavior observable in the aggregate are important behaviors shared by subsets of nodes in the system. We apply clustering to identify these behaviors. These characterizations provide a fuller picture of uptime behaviors present in PlanetLab.

In Section II, we describe the methods with which we analyzed PlanetLab trace data and in Section III we present the results of our analysis. In Section IV, we discuss our findings. In Section V, we discuss related work and we conclude in Section VI.

II. Methodology

A. PlanetLab Trace Data

Our objective in studying PlanetLab is to characterize the system in terms the behavioral patterns of uptime exhibited by its member nodes. For a single node, we define a continuous period of responsiveness as an *uptime*. We refer to the general profile of a node's observed uptimes over time as the *node's uptime behavior*. By extension, we make generalizations over the collective set of nodes' aggregated profiles and refer to these as the *system's uptime behavior*. To characterize PlanetLab in this manner, we need historic information that will allow us to approximate the uptimes of all participating nodes over a continuous period of time.

The data we used for our analysis comes from a collection of availability traces [6]. Within this collection, the particular dataset we used is called pl-app.avt, which shows the availability of nodes in PlanetLab from January 2004 through June, 2005. The number of active nodes over this duration ranges from 200 to 400, with individual nodes being identified by their IP addresses. There are 720 unique addresses contained in the set, of which 669 have at least one uptime (assumedly, some of the nodes were never up). For each of these nodes, the dataset contains a timeline indicating when the node was up and when it was down.

The timelines contained in the dataset were formed by observing pings sent between each pair of nodes at 15 minute intervals. At each interval, a node was considered up if it responded to at least half of the pings sent to it.

B. Analysis Tool

To analyze the uptime data, we built a trace-analysis tool that, among other things, produces a matrix that indicates how many of a given node's uptimes fall within various ranges of time (or buckets). This essentially characterizes the nodes with a set of traits, signaled by the buckets, such that the value in the mapping (number of uptimes) for a given bucket and a given node is the degree to which the node displays the particular trait.

To identify common patterns of behavior among nodes, we analyzed the described matrices using Cluto [11]. Cluto is a statistical analysis program developed to handle clustering tasks. It takes several tuning parameters and a characteristic mapping as input and runs a clustering algorithm over the data. It outputs element clusters, which in our case means sets of nodes characterized by similar patterns of uptime behavior. Cluto also produces information about the characteristics that most strongly bind the members in a given group to one another, as well as those that differentiate the group as a whole from other groups.

1060-9857/11 $26.00 © 2011 IEEE
DOI 10.1109/SRDS.2011.32

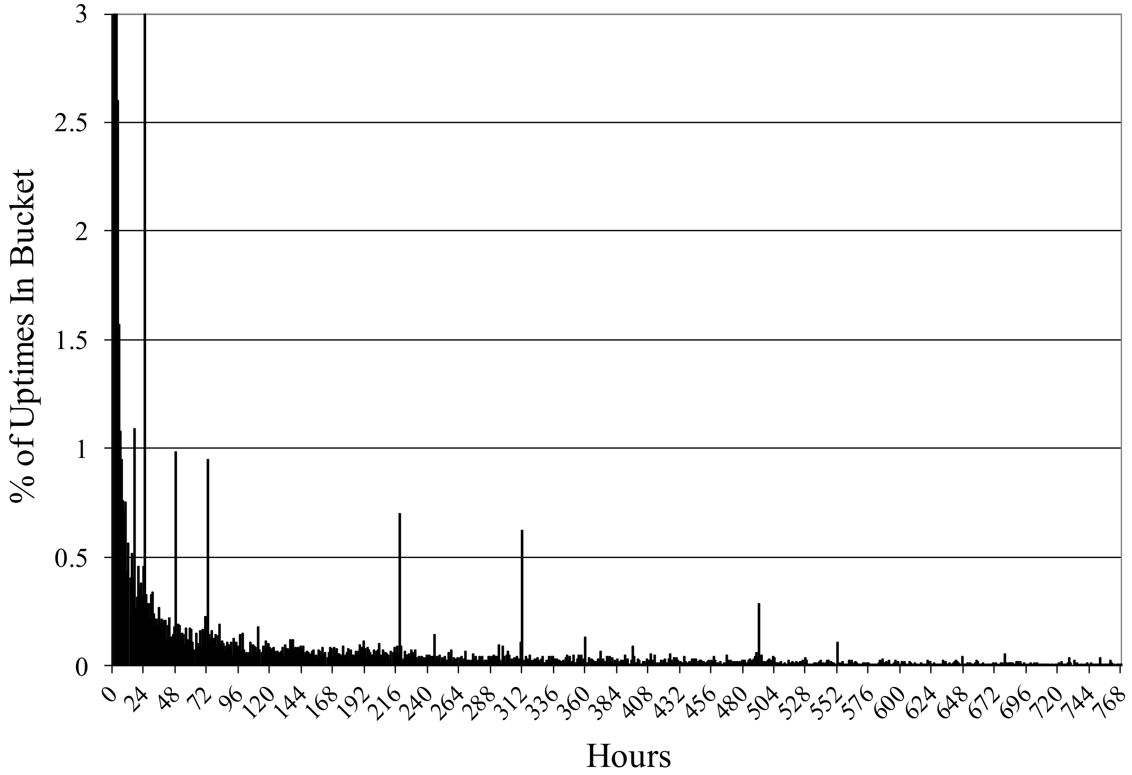


Figure 1. Histogram of uptimes in 1-hour buckets.

C. *Characterizing PlanetLab Nodes*

Using the described tools, we were able to cluster the nodes captured in the trace data by their uptime behaviors. From the resulting clusters, we determined the key behavioral characteristics that define the classifications. In the next section, we discuss the application of this classification process to the PlanetLab trace data.

III. Results

A. *Aggregate Distribution*

Fig. 1 shows the observed distribution of uptimes up to 32 days over the entire set of data. This includes 27447 (95.6%) of the total 28705 uptimes. Each bucket covers 1 hour. Looking at the chart, it does appear to bear a resemblance to standard distributions. To test this, we performed Chi-Square goodness-of-fit tests [17] to check whether the data could be modeled as Exponential or Weibull distributions.

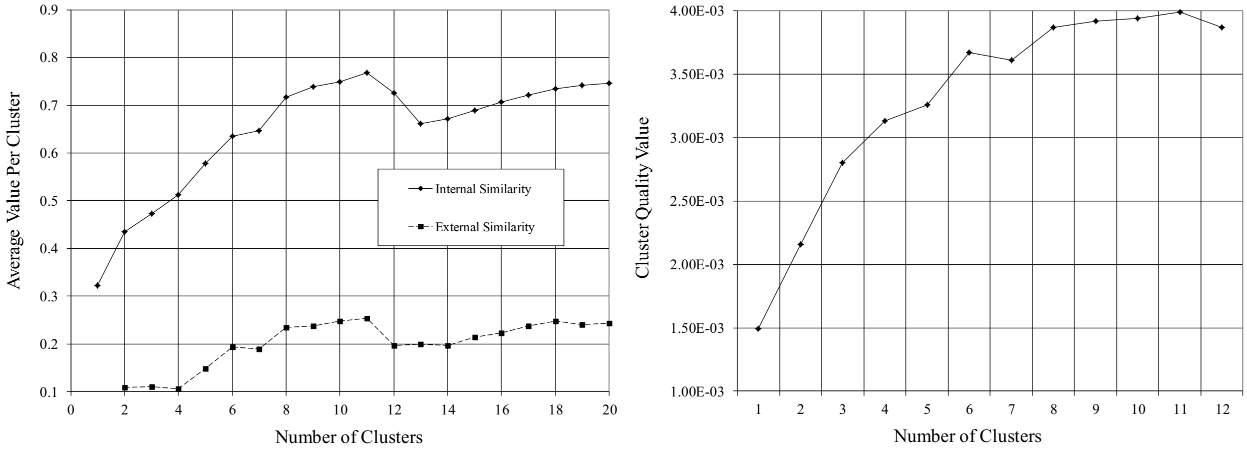
Starting with the Exponential distribution, we calculated a Chi-Square statistic value of 247142, where a good fit with a significance level of 5% requires the statistic to have been less than 42.6. Evaluating the Weibull distribution resulted in a Chi-Square statistic value of 21265. The parameters for the Weibull distribution were estimated using maximum likelihood estimation with initial values calculated using least-squares linear regression.

The biggest sources of difference between the observed and expected data are due to the spikes visible in Fig. 1. For example, there are 6815 entries in the first bucket (i.e., covering hour 1, which is the first hour), where the expected count using the Exponential distribution is about 333 and using the Weibull distribution is about 5615. At 24 hours, we observed 868 entries where Exponential expected about 251 and Weibull expected about 156. The number of instances of such behavior plus the degree to which the behaviors express themselves in each instance is an indication that characterization of PlanetLab should include such information.

Given this behavior, it may seem reasonable to characterize the system with an aggregate like the one presented here. However, there is much additional information to be gained by investigating the unique behaviors shared by subsets of the nodes being modeled, as we will see in Section III.B.

B. *Forming Clusters*

A single aggregate is a special case of clustering where the number of clusters produced is 1. It is possible that this adequately characterizes the target system. However, by looking a little deeper, we can find additional information



Figures 2 and 3. These graphs show how the quality of clustering changes with the number of clusters formed, where quality is positively affected by high similarity of nodes within a single cluster and negatively impacted by high similarity between different clusters.

useful in characterizing the system. For PlanetLab, we did this by using Cluto to identify shared group behaviors within the system, treating observed uptimes as traits of the nodes.

To perform this cluster analysis, we configured Cluto to use (1) a direct clustering algorithm, (2) the correlation coefficient as a measure of similarity, and (3) maximization of the ratio of intra-cluster similarity to inter-cluster similarity as a measure of the clustering quality (for more information on this cluster quality measurement, see "H2" in the Cluto documentation [11]).

When we performed our initial analysis, we found that most of the buckets were of little consequence to the outcome, so we trimmed the set of buckets to those that were identified as important in producing the clustering plus a few buckets that form spikes in the aggregate chart to focus the analysis. For the same reason, we also merged a few of these important buckets to form one trait (in each case, the buckets were adjacent time-wise). For example, we treat the hour 3 and hour 4 buckets as one 2-hour bucket. This resulted in 15 buckets including hour 1, hour 2, hours 3-4, hours 7-9, hour 16, hours 22-23, hour 24, hour 48, hour 72, hour 117, hour 191, hour 218, hour 312, hour 492, and hour 679.

Using this small set of significant buckets, we performed a number of clustering runs, for which we configured Cluto to form an incrementally increasing number of clusters in each proceeding run.

C. *Isolation of Behaviors*

Fig. 2 shows the internal similarity and external similarity for each of the runs we performed. Internal similarity is the similarity of uptime distributions among nodes grouped into a common cluster and external similarity is the similarity between different clusters. As described in Section III.B, similarity is measured using the correlation coefficient. Overall, this figure provides evidence that for this dataset, dividing nodes into clusters isolates unique shared behaviors.

Looking at Fig. 2, we find that internal similarity starts out increasing quickly, begins leveling out, dips, and finally resumes leveling out. For small numbers of clusters, adding an additional cluster has a strong impact, while the value of additions decreases as the total number of clusters starts to get large. This is partially explained by the fact that internal similarity is already high as we get into larger numbers of clusters. It is also a result of the new cluster not having a significant impact on increasing intra-cluster similarity or reducing intra-cluster similarity.

Unlike the internal similarity measurement, external similarity is flat up to 4 clusters. This reinforces the observations around the high impact of adding additional clusters when the total number of clusters is small. Low external similarity means that the behaviors exhibited by the different clusters are unique to those clusters. This combined with high internal similarity means that adding clusters is isolating unique behaviors that are strongly shared by cluster members.

Fig. 3 charts Cluto's internal measurement of overall quality. As described in Section III.B, quality is assessed using the ratio of internal to external similarity. While Fig. 2 shows a drop in internal similarity at 12 clusters, the overall quality measurement increases. This is due to the corresponding drop in external similarity, which increased quality enough that the drop in internal similarity was more than negated.

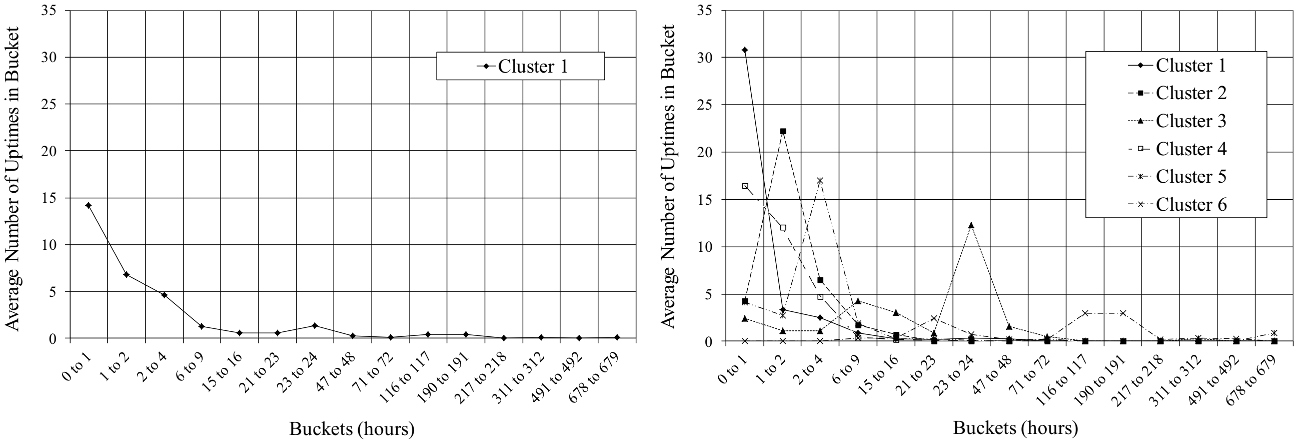
Looking at these charts, the six-cluster run roughly forms a knee in both and appears to be a good potential choice for a strong representation of significant behaviors within the system.

D. *Cluster Characteristics*

Figs. 4 and 5 respectively show the degree to which each trait (i.e., the buckets we selected) influences the different

TABLE I
NODE COUNTS OF THE CLUSTERS GENERATED DURING ANALYSIS OF PLANETLAB DATA

Number of Clusters	Sizes of Clusters	Mean Size (%)	Std. Dev. (%)
1	669 (100%)	100	-
2	390 (58%), 279 (42%)	50	11.31
3	339 (51%), 189 (28%), 141 (21%)	33.3	15.70
4	319 (48%), 56 (8%), 184 (28%), 110 (16%)	25	17.40
5	280 (42%), 90 (14%), 56 (8%), 141 (21%), 102 (15%)	20	13.13
6	218 (33%), 98 (15%), 77 (12%), 61 (9%), 114 (17%), 101 (15%)	16.7	8.40
7	210 (31%), 89 (13%), 55 (8%), 74 (11%), 111 (17%), 38 (6%), 92 (14%)	14.3	8.24
8	77 (12%), 184 (28%), 63 (9%), 54 (8%), 41 (6%), 57 (9%), 93 (14%), 100 (15%)	12.5	6.93
9	77 (12%), 178 (27%), 61 (9%), 25 (4%), 54 (8%), 29 (4%), 52 (8%), 93 (14%), 100 (15%)	11.1	7.09



Figures 4 and 5. These show the behavioral characteristics of clusters formed by the single-cluster run and the six-cluster run, respectively. Observe that Cluster 1 has a similar shape in each chart, though since nodes with dissimilar behavior have been removed from Cluster 1 in the six-cluster run, the average number of uptimes in the 1-hour bucket increases. Likewise, the other clusters in the six-cluster run show behaviors indiscernible in the single-cluster run.

clusters for the one- and six-cluster runs. Observe that in the six-cluster run, there are six distinct patterns of behavior. In both figures, Cluster 1 has some similarity of shape, but the removal of dissimilar nodes in the six-cluster run has resulted in an increase in behavioral cohesion, resulting in some traits showing greater influence in the cluster and others showing less.

Although we omit additional charts for brevity, visual observation of the charts for the different runs concurs with the observations made in Section III.C. That is, increasing the number of clusters beyond a point yields diminishing returns in terms of the uniqueness of the behavioral patterns of uptime.

Table I provides additional information that is not captured by the quality indictors or behavioral patterns. It shows both the absolute and relative sizes of clusters formed in runs 1 through 9, where each run is done with an increasing number of clusters (i.e., the target number of clusters for run *n* is *n*). This table shows that beyond six clusters, adding additional clusters generally results in several clusters that are small in size relative to the mean. This is an indication that the behaviors in the additional clusters are quite specific and applicable to only a small set of nodes. Observe in Table I that the smallest cluster in the 7-cluster run (38 nodes) is 33% smaller than the smallest cluster in the 6-cluster run (61 nodes). The 7-cluster run's smallest cluster is also around 42% of the mean cluster size for that run, whereas the 6-cluster run's smallest cluster is around 54% of the respective mean.

IV. Discussion

The observations made in Section III show unique behavioral patterns that are useful for the characterization of PlanetLab. Since these patterns represent real behaviors, they provide a more accurate model of how the system behaves.

Having a good understanding of the behavioral characteristics of uptime for a system like PlanetLab is important for performing analysis, supporting proper app-level design decisions, and helping to explain certain events. The lack of such a model forces one to rely on assumptions that may not be correct and can cause the application to fail expectations.

Since many aspects of distributed applications are designed around handling failure, having this level of detail in the model is often quite important. For example, consider a distributed file system that uses redundancy to ensure data availability. If node uptime is generally expected to follow a standard distribution, designers would miss the effects of the spikes we see in the aggregate in Fig. 1. Further, they would miss the substantial differentiation of subsets of nodes shown in Fig. 5. The result of this may be the over- or under-maintenance of data copies. Over-maintenance is costly, while under-maintenance puts one's service level agreements at risk.

To build a more concrete example, we consider CoDeeN, a decentralized CDN built on PlanetLab [20]. The system is designed such that nodes avoid "problematic" peers rather than incurring the performance hit those nodes would impose. The question of determining peer group size, which is based on the stability of peers, is labeled as an open question. Based on our observation of behaviors within PlanetLab, this future work would benefit from accounting for the differences in subsets of nodes within the system. That is, the differences in subsets of nodes can allow for more refined tuning of peer-selection and perhaps pro-active changes in the topology based on the expectation of pending failure.

Aside from designing and tuning applications around the behavior of PlanetLab itself, the behaviors we have identified can also help to identify groups of nodes that reflect the behavior of the target environment for which one is creating the application. For example, if the application is to be deployed to an environment in which nodes are highly stable, it makes more sense to test the application on the nodes that most closely reflect this behavior than on a random selection of nodes.

V. RELATED WORK

In [9], the authors investigated methods of classifying failures and studied node behaviors in the PlanetLab test bed. Although they didn't use clustering techniques, they devised a systematic method of classifying failures by duration, group-size, and the nature of the failure. They further apply additional information such as geography, error messages, and resource usage to explain possible causes of the errors.

In [10], the authors studied classification of the availability of CPUs in a massive distributed system through the use of clustering techniques. In their analysis, they excluded all nodes with nonrandom behavior to focus on emergent patterns across nodes whose behaviors are independent and identically distributed. Interestingly, they too found six clusters to be the right number for behavioral classification. In their analysis, they focused on subdividing the nodes into groups whose behavior could be modeled using known statistical distributions.

Several works have presented other unique perspectives on representation of node uptimes. In one such study, the authors propose a method to model system-wide churn behavior that arises from individual node behavior, but does not require modeling the behavior of individual nodes [12]. In another, the authors find three noticeable, distinct behaviors in a dataset containing node downtimes measured on a large corporate network [2], where each of these behaviors can be represented by a well-known distribution. Using this insight, the authors formed a unique downtime model that was composed of a combination of the three distributions

Various works have also shown the different models suitable for representing component uptimes in different systems and environments. Examples of the variety of systems that have been studied include a P2P file-sharing system [3], student labs at universities [8][14], worker nodes in a small resource pool [14], Internet hosts [14], and high-performance computing clusters [18][19]. These studies show repeatedly that the most suitable model of system behavior varies greatly on the system in question.

A number of systems use knowledge of expected uptimes to make functional decisions. One such system is the Total Recall storage system [1]. Total Recall uses the availability of participating nodes to determine the amount of data redundancy needed to maintain a specific level of availability for the data. To make replication decisions, live measurements of short-term node availability are feed into the redundancy policy, which estimates the likelihood of a set of hosts being available at a given time. In [5], the authors suggest the alternative of quantifying short-term node availability from the viewpoint of the system's participants rather than from the viewpoint of an outsider. The authors show that the number of file transfers required under this alternative scheme is as little as 25% of what was originally thought to be needed to meet the data-availability goal. This seems to be strong evidence of how having a different, perhaps more accurate, model of availability can provide significant benefit.

Instead of modeling availability, it is also possible to simulate it. In [7], the authors present DieCast, which describes a way in which to simulate huge systems in their entirety with only a fraction of the resources needed for actual deployment. The trade-off, however, is that a large amount of time is spent in simulation since the scale-down in resource consumption is achieved through time-dilation.

The authors of [13] explore the state-machine approach to characterizing node uptimes. Instead of using probabilistic models, they attempt to predict uptimes and downtimes using state machines based on branch predication techniques.

VI. CONCLUSION

By applying a Chi-Square goodness of fit test to an aggregation of PlanetLab node uptime data, we found the distribution of this aggregation to be poorly fitted to the Exponential and Weibull distributions. We observed that visible aberrations in the aggregate distribution of node uptimes were a primary cause of this.

Using clustering techniques, we iterated through the formation of different numbers of clusters to identify a clustering of high quality that does not form arbitrary groupings as occurs when too many cluster are allowed to form. This led us to a set of six disjoint groups of nodes, in which each group is characterized by a distinct pattern of uptime behavior.

We believe this set of clusters to be valuable to those building on PlanetLab and perhaps to the designers of PlanetLab itself. Knowledge of the behaviors exhibited by the clusters can contribute to making better design decisions and ensuring that testing is being performed under valid assumptions.

Future work around gathering additional information on uptime behaviors in distributed systems includes:

- Analysis using different interpretations of responsiveness (different ping requirements, CPU availability, etc.)
- Analysis of downtime
- Analysis of different systems

It would also be interesting to see how the behavioral patterns of clusters would change if the clusters were not disjoint. Finally, it goes without saying that we would like to apply our findings to the design and tuning of real systems and applications.

REFERENCES

[1] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. M. Voelker. Total Recall: System support for automated availability management. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, 2004.

[2] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In *ACM SIGMETRICS Performance Evaluation Review*, Volume 28, Issue 1, 2000.

[3] F. E. Bustamante and Y. Qiao. Friendships that last: Peer lifespan and its role in P2P protocols. In *Web Content Caching and Distribution: Proceedings of the 8th International Workshop*, 2004.

[4] A. Datta, K. Aberer. Internet-scale storage systems under churn - A study of the steady-state using Markov models. In *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, 2006.

[5] R. J. Dunn, J. Zahorjan, S. D. Gribble, and H. M. Levy. Presence-based availability and P2P systems. In *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*, 2005.

[6] B. Godfrey. Repository of availability traces. Available online at *http://www.cs.illinois.edu/~pbg/availability/*.

[7] D. Gupta, K. Vishwanath, and A. Vahdat. DieCast: Testing distributed systems with an accurate scale model. In *Proceedings of the 5th ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[8] T. Heath, R. P. Martin, and T. D. Nguyen. The shape of failure. In *Proceedings of the First Workshop on Evaluating and Architecting System dependabilitY (EASY)*, 2001.

[9] S. Jain, R. Prinja, A. Chandra, and Z. Zhang. Failure classification and inference in large-scale systems: A systematic study of failures in PlanetLab. *Technical Report 08-014, University of Minnesota - Computer Science and Engineering*, 2008.

[10] B. Javadi, D. Kondo, J. Vincent, and D. P. Anderson. Mining for statistical models of availability in large-scale distributed systems: An empirical study of SETI@home. In *Proceedings of the IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2009.

[11] G. Karypis. CLUTO – A clustering toolkit. *Technical Report #02-017, Department of Computer Science, University of Minnesota*, 2003.

[12] S. Y. Ko, I. Hoque, and I. Gupta. Using tractable and realistic churn models to analyze quiescence behavior of distributed protocols. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2008.

[13] J. W. Mickens and B. D. Noble. Exploiting availability prediction in distributed systems. In *Proceedings of the 3rd Symposium on Networked Systems Design & Implementation (NSDI)*, Volume 3, 2006.

[14] D. Nurmi, J. Brevik, and R. Wolski. Modeling machine availability in enterprise and wide-area distributed computing environments. In *Proceedings of European Conference on Parallel Computing (EUROPAR)*, 2005.

[15] PlanetLab – An open platform for developing, deploying, and accessing planetary-scale services. Available online at *http://planet-lab.org/*.

[16] S. Ramabhadran and J. Pasquale. Analysis of long-running replicated systems. In *Proc. of the 25th IEEE Annual Conference on Computer Communications (INFOCOM)*, 2006.

[17] J. L. Romeu. The Chi-Square: A large-sample goodness of fit test. *RAC START*, Volume 10, Number 4, 2004.

[18] B. Schroeder and G. A. Gibson. A large-scale study of failures in high-performance computing systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2006.

[19] B. Schroeder and G. A. Gibson. Disk failures in the real world. In *Proceedings of the 5th USENIX conference on File and Storage Technologies*, 2007.

[20] L. Wang, K. Park, R. Pang, V. Pai, and L. Peterson. Reliability and Security in the CoDeeN Content Distribution Network. In *Proceedings of the USENIX 2004 Annual Technical Conference*, 2004.