Statistical Modeling of Resource Availability in Desktop Grids *

Joshua Wingstrom Henri Casanova

Information and Computer Sciences Dept.
University of Hawai'i at Manoa, Honolulu, U.S.A.

Abstract

Desktop grids are compute platforms that aggregate and harvest the idle CPU cycles of individually owned personal computers and workstations. A challenge for using these platforms is that the compute resources are volatile. Due to this volatility the vast majority of desktop grid applications are embarrassingly parallel and high-throughput. Deeper understanding of the nature of resource availability is needed to enable the use of desktop grids for a broader class of applications. In this document we further this understanding thanks to statistical analysis of availability traces collected on real-world desktop grid platforms.

1. Introduction

Desktop grids are platforms that exploit the idle CPU cycles of distributed and typically individually owned computing resources, e.g., desktop computers. Many desktop grid systems have enabled *volunteer computing* [22] on the Internet, as in the famous SETI@home project [25] and others [16][8]. In such systems the incentive for users to participate is because the target application is deemed worthwhile by resource owners. Desktop grids have also been deployed successfully in enterprise environments, e.g., an organizations' local area network to execute applications that are of interest to that organization. Several desktop grid infrastructures are available from academia [16, 10, 2] and industry [27]. Although some of these infrastructures vary in usage they all use computing resources that are volatile. As a result, the overwhelming majority of applications executed on these platforms consist of large number of tasks (relatively to the number of computing resources), which we term "high-throughput" applications. For these applications, a good metric for performance is the rate at which tasks complete per time unit, and a simple first-come-first-serve policy to assign tasks to idle hosts is effective.

Several researchers have explored ways in which desktop grids can be used to run applications other than high-throughput applications. For instance, the work in [6] provides a way to execute applications in which tasks can synchronize and communicate using MPI [24]. However, the difficult question of which task should be assigned to which resource(s) for best performance and/or highest probability of application completion remains. This scheduling question has been investigated in [13] for applications that depart from the high throughput model due to a small number of tasks (relatively to the number of computing resources). Not surprisingly the results in [13] indicate that scheduling should account for the availability of the underlying resources. The authors show empirically that combining simple resource exclusion, resource prioritization, and task duplication techniques lead to good application performance. The authors also attempted to use some amount of information about past availability of the underlying, namely the number of CPU cycles delivered during the previous day, which unfortunately proved ineffective. In this paper we attempt to provide a deeper statistical characterization of resource availability, which we argue is key to improve existing resource selection and task duplication strategies. Such a characterization would represent a fundamental basis on which to develop (more) effective strategies for executing non-high-throughput applications on desktop grids.

Statistical characterization of desktop grid resource availability must rely on statistical analysis of availability measurements collected on real-world systems. Fortunately, several such datasets have been collected in previous work, some as recently as 2005. While these datasets can be used for driving simulations of desktop grids, as for instance in [13], in this paper we are interested in using them for statistical analysis. Two recent articles have an-

alyzed availability datasets: Nurmi et al. [20] and Kondo et al. [14]. The former provides an in-depth statistical analysis of host availability intervals, that is the durations in seconds of times between host failures. The latter provides a high-level analysis of CPU availability in terms of actual CPU cycles delivered to a hypothetical desktop grid application between two application task failures. Our contributions, which are based on datasets used in [20] and/or in [14], are:

1. Although it is not mentioned in [20], we found that the Log-Normal distribution is a good candidate for modeling resource availability.
2. We present analysis for numbers of CPU cycles delivered to a desktop grid application, in addition to just seconds of availability as in [20], which makes our results more meaningful to desktop grid applications. Inspired by the work in [20] however, we use detailed statistical modeling. We thus gain more in-depth understanding of resource availability than in [14].

This paper is organized as follows. In Section 2 we provide background on resource availability datasets and discuss previous work. Section 3 details our statistical analysis of the datasets and highlights our main results regarding the distribution of availability. Section 4 concludes with perspectives on future work.

2. Related Work

Our results are obtained through analysis of desktop grid availability datasets that have been obtained over the years. Availability measurements in these datasets are collected in a variety of ways and for a variety of metrics. [14] contains a useful discussion of what an "ideal" dataset should contain for the purpose of studying the behavior and performance of desktop grid applications, which we recall here briefly. Three types of resource availability are considered at each instant: (i) *host availability*, that is whether the resource is up or down; (ii) *task execution availability*, that is whether a desktop grid application task could potentially run on the resource at that instant (this may not be the case if, e.g., the user uses the mouse of keyboard); and (iii) *CPU availability*, that is the fraction of the CPU cycles that the task would be able to use. And ideal dataset would capture all three types of availability, provide full information on the configuration of each measured resources, and identify causes for unavailability. Several host availability datasets have been collected [17, 5, 7, 23]. Based on such availability data, one can define the concept of *availability intervals* on a resource, that is the number of seconds or of CPU cycles in between two observations of host/task unavailabilities.

Our work is strongly related to two recent research articles that have analyzed desktop grid resource availability datasets. In Nurmi et al. [20], the authors analyze three datasets, two task execution availability datasets and one host availability dataset. Using sound statistical analysis techniques, the authors conclude that a Weibull or a Hyper-exponential distribution models resource availability reasonably well, with a strong preference for the Weibull distribution due to its simplicity. Importantly, this work dispels the notion that using the classic Exponential or Pareto distribution is appropriate. In this paper we reproduce some of the results in [20] and show that the Log-Normal distribution is actually a better candidate than the Weibull distribution since it is as simple and often a better fit. The bulk of our results however targets CPU availability datasets, which are arguably more relevant to study an execution of a desktop grid application. It is conceivable that CPU availability could be inferred from the datasets used in [20]. However, based on the discussion in [14] and on the fact that datasets that do measure CPU availability directly are available, we opted to use these datasets instead.

In [14], four CPU availability datasets are discussed and broad characterizations of each datasets are presented. Three of these datasets were obtained via an "active measurement" technique, that is by actually running a desktop grid application on a desktop grid infrastructure and measuring how many CPU cycles that application was able to use on each host (accounting for keyboard/mouse activities and desktop grid software configurations). The fourth dataset is not a CPU availability dataset per se, but rather consists of CPU, keyboard, and mouse activity data collected in [3]. From this data the authors in [14] were able to reconstruct approximate CPU availability data. In this paper we use all four availability datasets.

Ultimately, our work is related to works that investigate the deployment of non-high-throughput applications on desktop grid resources [13, 18, 19, 9, 11, 4, 1, 15]. Indeed, better statistical understanding of desktop grid resource availability provides a fundamental basis on which to develop (more) effective strategies for scheduling and executing many classes of applications on desktop grids.

3. Statistical Analysis of Availability Datasets

3.1. Availability Datasets

In this work, we analyze eight datasets collected from real desktop Grid platforms. Three of these datasets are analyzed for availability in both operations completed and seconds available. The traces where gathered by submitting measurement tasks to a desktop Grid system. These measurement tasks are perceived and executed as real tasks. They perform computation and periodically write their computation rates to file.

The CSIL data set is from UCSB's CSIL computer science student lab. Each measurement records the time from when a workstation is able to run a user process to when it no longer can do so. The Condor [26] dataset is from the Condor pool running at the University of Wisconsin. Condor availability measurements are created by measuring the time from when the Condor scheduler starts one of the monitoring processes to when the allocated workstation evicts the monitoring process. The CSIL dataset is from 83 machines measured for 8 weeks. The Condor dataset comes from 210 machines measured for a 6-week period.

The UCB trace originated from an older data set first reported in [4]. The traces were collected using a daemon that logged CPU and keyboard/mouse activity every 2 seconds over a 46-day period (2/15/94-3/31/94) on 85 hosts. The hosts were used by graduate students in the EE/CS department at UC Berkeley. These traces were then processed to reflect the availability of the hosts for a desktop Grid application [12].

The Entropia trace originated from running the commercial Entropia desktop Grid software on desktop PC's at the San Diego Super Computer Center (SDSC). During a cumulative 1 month period in the last quarter of 2003, availability measurements were conducted with the deployment of Entropia DCGrid over about 200 hosts. Their clock rates ranged from 179 MHz up to 3.0 GHz, with an average of 1.19 GHz. One problem with the Entropia trace is that the Entropia system makes use of a Virtual Machine (VM) to insulate application tasks from the resources. The problem is that a bug in the VM caused the trace to be missing some data. The traces were generated by repeatedly computing limited length tasks. However, when a failure would occur in the system, the trace would only be recorded up until the end of the last task. This has caused the traces to be a bit pessimistic and to have some periodicity.

The XtremWeb traces were collected using the XtremWeb desktop grid software continuously over about a 1 month period (1/5/05 - 1/30/05) on a total of about 100 hosts at the University of Paris-Sud. In particular, XtremWeb was deployed on a cluster with a total of 40 hosts, and a classroom with 40 hosts. The cluster is used by the XtremWeb research group and other researchers for performance evaluations and running scientific applications. The classroom hosts are used by first year students.

3.2. Distribution of availability intervals

The first natural step when trying to understand the nature of resource availability is to fit the empirical distribution of availability intervals to known distributions. This was done in previous work [20] for two of the data sets we use in this work, Condor and CSIL, which measure host and/or task execution availability. The results were that both datasets were fit about equally well by the Hyper-exponential distribution and only the CSIL dataset was a good fit for the Weibull distribution.

In this section we make two additional contributions. First, we use 5 other data sets, for a total of 8. And for these 5 additional datasets we also study the distribution of CPU availability intervals. Second, we evaluate our presumption that the Log-Normal distribution, which was left out of the study in [20], would in fact be a good candidate. In this paper we only present results for the Weibull and the Log-Normal distribution, while in [20] authors also studied the Hyper-exponential distribution. Given the nature of the Hyper-exponential distribution, we can assume it would fit all datasets well, but would bring little benefit, given that it is more complex and thus more difficult to incorporate in analytical models. Therefore, it is not discussed in this paper.

3.2.1. Graphical evaluation of fits We first investigate the quality of the fits for each dataset graphically. The empirical Cumulative Distribution Function (CDF) of the dataset is plotted on the same graph as the known CDFs of the two candidate distributions, Weibull and Log-Normal. The parameters of this distribution were estimated from the sample dataset using Maximum Likelihood Estimation (MLE), and are shown in Table 2. The three CDFs are compared visually to determine how closely the candidate distributions track the data.

Name	Number of Hosts	Duration	Host Availability	CPU Availability
Condor	210	6 weeks	√	
CSIL	83	8 weeks	√	
UCB	85	46 days	√	√
Entropia	232	1 month	√	√
XtremWeb C	30	25 days	√	√
XtremWeb Linux	136	25 days	√	√
XtremWeb PC	7	25 days	√	√
XtremWeb Sim	55	25 days	√	√

Table 1. Data Set used for Fitting

	Units	Weibull		Log-Normal	
		α	β	μ	σ
Condor	seconds	0.5497	2735	7.103	1.701
CSIL	seconds	0.5537	$2.827e^5$	11.49	2.218
UCB	operations	0.3512	$4.29e^7$	16.42	2.419
	seconds	0.3551	443.7	4.807	2.419
Entropia	operations	0.4781	$4.0e^8$	18.76	2.188
	seconds	0.5495	$1.319e^4$	8.646	1.764
XtremWeb C	operations	0.516	$1.866e^9$	20.33	2.132
	seconds	0.5191	$1.322e^4$	8.483	2.109
XtremWeb Linux	operations	*no-fit*	*no-fit*	25.04	11.3
	seconds	*no-fit*	*no-fit*	*no-fit*	*no-fit*
XtremWeb PC	operations	0.4203	$1.109e^9$	19.55	2.66
	seconds	0.4303	$1.365e^4$	8.287	2.577
XtremWeb Sim	operations	0.1724	$3.742e^{10}$	22.95	2.926
	seconds	*no-fit*	*no-fit*	*no-fit*	*no-fit*

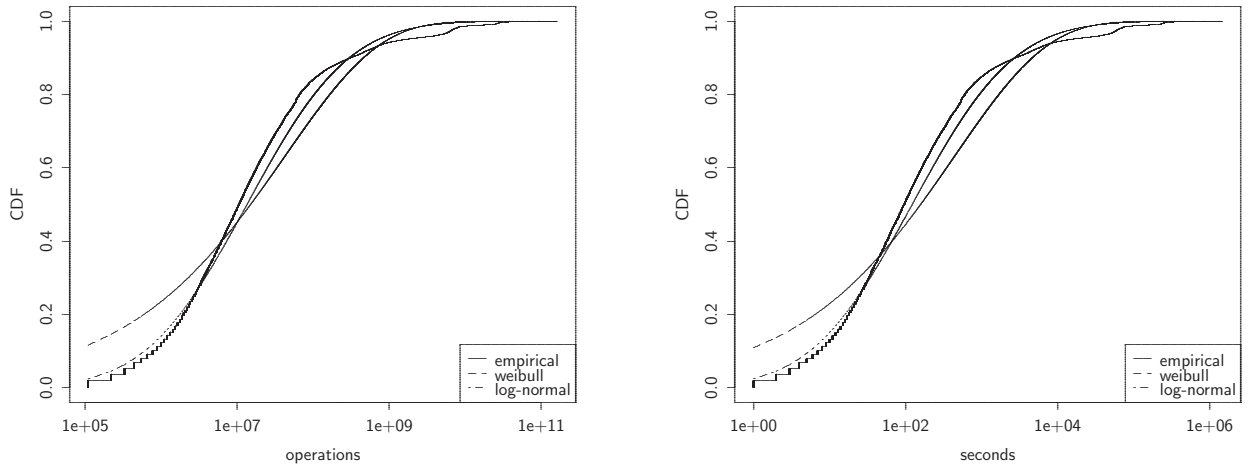
Table 2. Weibull and Log-Normal parameter values obtained via MLE, for all datasets in terms of seconds and number of operations.

Figure 1 and Figure 2 show the graphical examinations of the condor and CSIL datasets respectively. For these CDF plots and all other CDF plots presented in this paper, we use a logarithmic scale for the x-axis. For the Condor dataset, we can see that both the log-normal and the Weibull distribution do not track the empirical distribution well (the fact that the Condor dataset is difficult to fit to a distribution is discussed in [20]). The Weibull distribution appears, from a graphical standpoint, to be a better fit for the CSIL dataset than the log-normal distribution.

The situation is reversed for the UCB dataset, shown in Figure 3. The UCB data demonstrates that the Weibull distribution has problems problems capturing the shape of the availability intervals on the UCB data. Initially, Weibull's predictions are to large. This problem is slowly alleviated until the CDFs meet, at which point the Weibull distributions errors on the opposite side. The log-normal distribution accurately tracks the empirical CDF for the first two-thirds of the x-axis. Both distributions are a small bit off for the last third of the x-axis, but Weibull does a little bit better here.

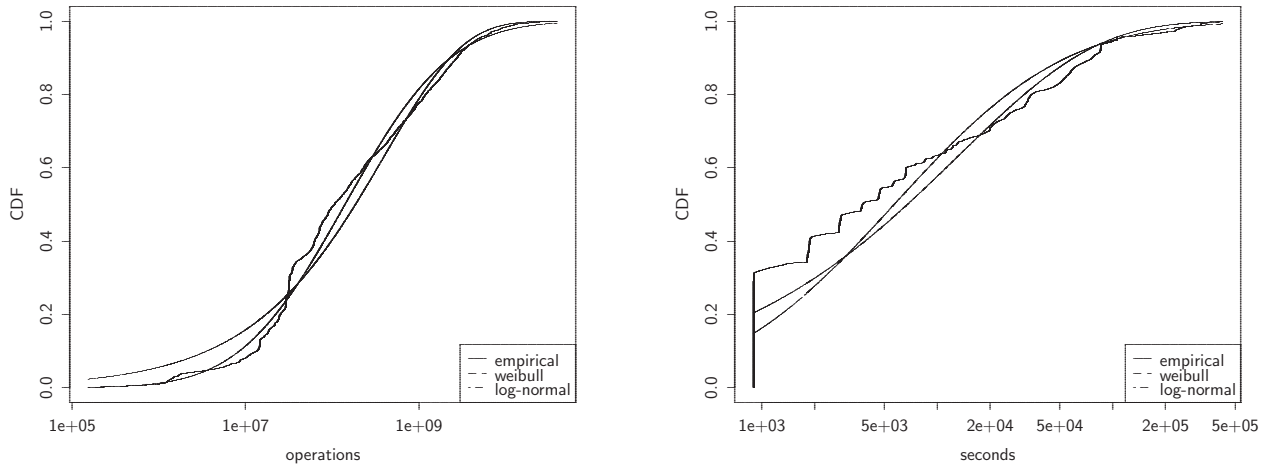
Figure 4 is similar to Figure 3 but is for the Entropia dataset. We observe a somewhat similar phenomenon as with the UCB dataset on the left-hand side of the graph for availability intervals in terms of operations. It is difficult to say which candidate distribution is best in the case of availability intervals in terms of seconds. As discussed in Section 3.1, the data suffers from periodicity and is thus at the same time difficult to fit and inacurrate.

Figures 5,6,7,8, show the CDFs for the



(a) Operations (b) Seconds

Figure 3. Empirical and fitted CDF of the availability interval length (in number of operations and seconds) for the UCB dataset.



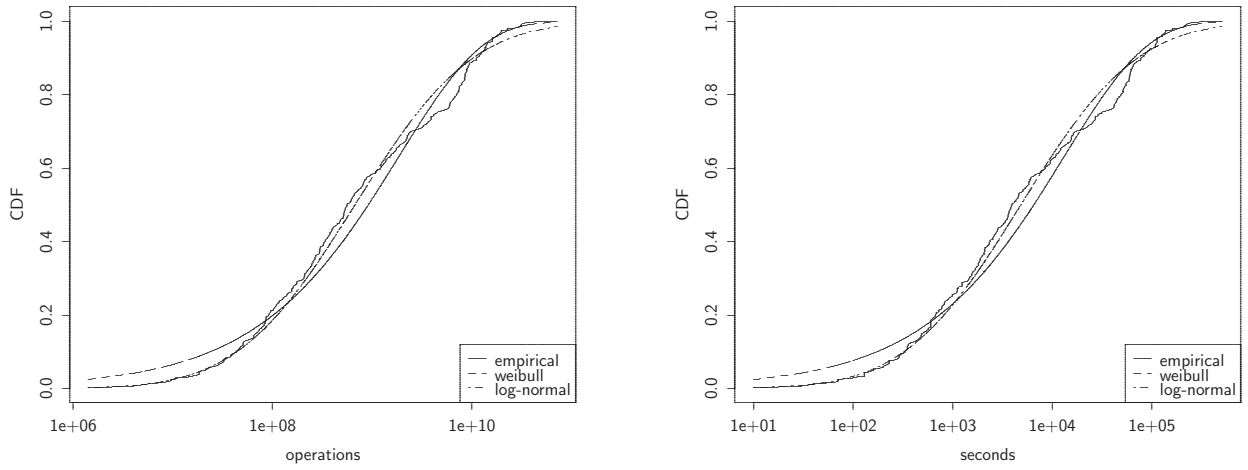
(a) Operations (b) Seconds

Figure 4. Empirical and fitted CDF of the availability interval length (in operations and seconds) for the Entropia dataset.

XtremWeb datasets.

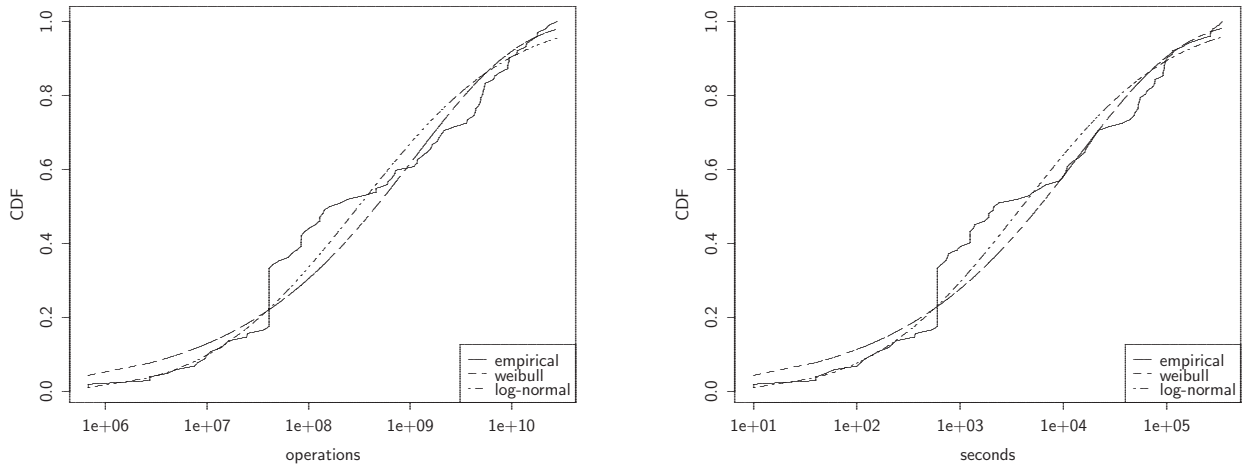
Again in Figure 5 we see that for the vast majority of the points, on the left-hand side of the graphs, the Log-Normal distribution does a much better job of fitting the distribution. Both distributions have difficultly capturing the group of intervals at $\sim 10^9$ operations and $\sim 10^4$ seconds.

On Figure 6 we can see that the XtremWeb PC dataset is much more "jagged" and difficult to fit. Nevertheless, here again the Log-Normal distribution does better than the Weibull distribution with the small values, again. However, the Weibull distri-



(a) Operations (b) Seconds

Figure 5. Empirical and fitted CDF of the availability interval length (in operations and seconds) for the XtremWeb C dataset.



(a) Operations (b) Seconds

Figure 6. Empirical and fitted CDF of the availability interval length (in operations and seconds) for the XtremWeb PC dataset.

bution seems to better account for the large amount of intervals that occur around 10^9 operations and 10^3 seconds.

Figures 7,8, show the CDFs for the XtremWeb Linux and XtremWeb Sim datasets. The recorded XtremWeb Linux and XtremWeb Sim datasets are not good candidates for either the Weibull or log-normal distributions. The data is so difficult to categorize for these distributions that our fitting algorithms could not compute reasonable values in many cases. The XtremWeb Linux data may be caused by too short of a trace. Since the data comes from a cluster, which, for a large most part, operates like a single computer, the system may need to run nearly

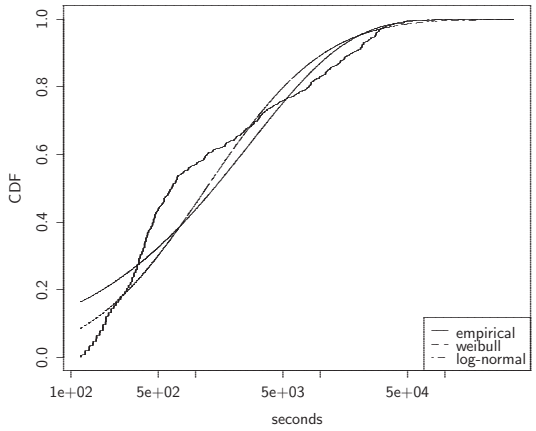


Figure 1. Empirical and fitted CDF of the availability interval length (in number of seconds) for the Condor dataset.

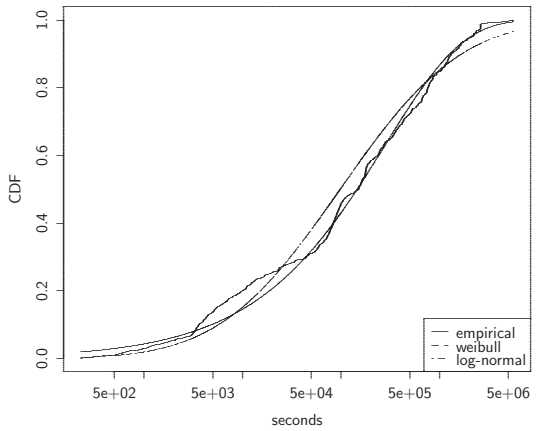


Figure 2. Empirical and fitted CDF of the availability interval length (in number of seconds) for the CSIL dataset.

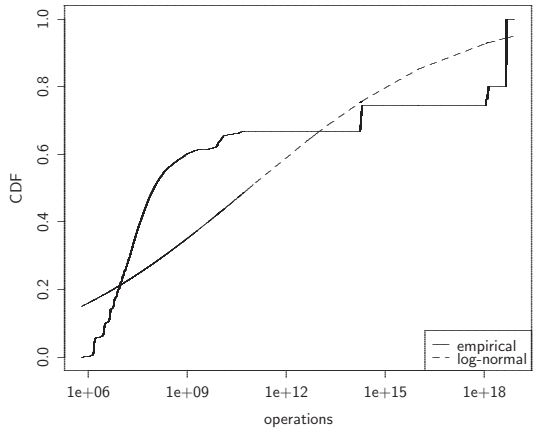


Figure 7. Empirical and fitted CDF of the availability interval length (in operations) for the XtremWeb Linux dataset.

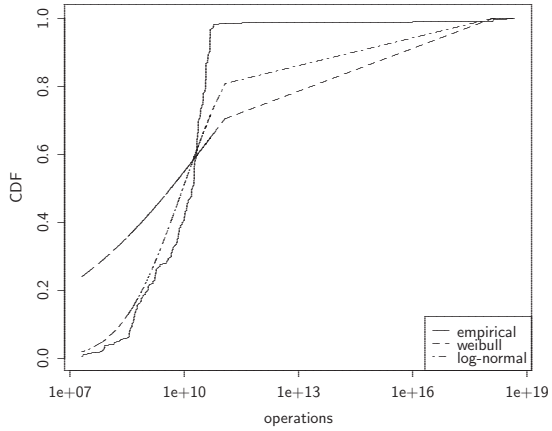


Figure 8. Empirical and fitted CDF of the availability interval length (in operations) for the XtremWeb Sim dataset.

as many times longer as there are other hosts in the other systems, for the trace to become as smooth. The XtremWeb Sim data is thrown off by a few very large values.

3.2.2. Numerical evaluation of fits To evaluate the quality of the fit we also use the Kolmogorov-Smirnov (KS) test, employing the classic procedure used in [20]. Briefly, we choose subsamples from the dataset of size 100 and compute the corresponding *p-value*. This is done 1000 times to obtain a range of values and the average test statistic value is used to compute the *p-value* for this iteration. The results from these 1000 iterations are not necessarily normally distributed. However, according to the Central Limit Theorem (CLT), if this experiment is repeated, the distribution of these averages is normally distributed. Therefore, 1000 subsamples of size 100 are computed 100 times to obtain a point estimation for the KS test *p-value*. KS test values are shown in Table 3.2.2.

The results of the KS test match with our graph-

	Units	Weibull		Log-Normal	
		p-value	$s_{p\text{-}value}$	*p-value*	$s_{p\text{-}value}$
Condor	seconds	0.003696	$6.006e^{-5}$	0.02723	0.001959
CSIL	seconds	0.3292	0.008131	0.156	0.005472
UCB	operations	0.02226	0.0007394	0.3249	0.008038
	seconds	0.02311	0.0009394	0.3252	0.008364
Entropia	operations	0.1226	0.003355	0.2337	0.006302
	seconds	0.0004406	$3.517e^{-6}$	0.009744	0.0002554
XtremWeb C	operations	0.1637	0.00492	0.2855	0.006479
	seconds	0.1668	0.005082	0.2935	0.007957
XtremWeb Linux	operations	*no-fit*	*no-fit*	$3.012e^{-5}$	$4.495e^{-6}$
	seconds	*no-fit*	*no-fit*	*no-fit*	*no-fit*
XtremWeb PC	operations	0.02756	0.0003061	0.1444	0.0007969
	seconds	0.04369	0.0006281	0.1669	0.001228
XtremWeb Sim	operations	$1.5e^{-9}$	$9.09e^{-11}$	$6.949e^{-6}$	$3.377e^{-7}$
	seconds	*no-fit*	*no-fit*	*no-fit*	*no-fit*

Table 3. KS Test P-Values

ical evaluation of the fits. The Weibull distribution works marginally better than the Log-Normal distribution for the Condor dataset, although neither does exceptionally well. Weibull shows a significant accuracy increase over Log-Normal with the CSIL dataset, but Log-Normal shows a near equal improvement over Weibull for the XtremWeb C dataset. Log-Normal does do exceptionally better on the UCB dataset, Entropia dataset and XtremWeb PC datasets. Both Weibull and Log-Normal distributions do a poor job of fitting the XtremWeb Linux and XtremWeb Sim datasets, as expected given our observations in Section 3.2.1.

The Weibull distribution's accuracy on the CSIL and Condor datasets is also published in [20]. Our results are similar. However, [20] reports slightly different fitting values (likely a result of variations in the optimization techniques used in MLE fitting) and higher *p-values*. This initially caused concern as to whether our algorithms had achieved a suboptimal fit. Investigation shows that these larger *p-values* come from using different methods for computing the KS-Test *p-values*. Our values were computed using the *ks.test* function in the stats package in R[21]. For an example of the differences, see Table 3.2.2.

4. Conclusion

We have provided results that show the log-normal is also, in addition to Weibull a good choice for modeling resource availability data. In many cases with the new data we provide, log-normal is a better choice for modeling availability data as it is particularly better at generating data points for small values. Additionally, the log-normal distribution excelled at modeling data that was collected in terms of operations, which is more useful for modeling desktop grid performance. Time available and operations delivered are different in desktop grids because application performance is throttled to avoid impacting the users. We have also shown, using the log-normal distribution and new datasets, that availability data can be modeled fairly accurately using relatively easy to work with models.

In the future we hope to examine how accurate a model needs to be in order for it to be useful for both simulations and for scheduling. We also hope to examine how these models can be used to make other inferences about the data. Also, with regard to modeling, these results do not take into account the stationary of the data and the correlation between hosts. Studies of these statistics would likely also prove useful.

References

[1] A. Acharya, G. Edjlali, and J. Saltz. The utility of exploiting idle workstations for parallel computation. In *Proceedings of the 1995 ACM SIGMETRICS Conference*, pages 225–234, May 1997.

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

[3] R. Arpaci, A. C. Dusseau, A. Vahdat, L. Liu, T. Anderson, and D. Patterson. The Interaction of Parallel and Sequential Workloads on a Network of Work-

	Units	Our Test		KS test in [20]
		p-value	$s_{p\text{-}value}$	*p-value*
Condor	seconds	0.0002324	$3.134e^{-6}$	0.07
CSIL	seconds	0.3379302	0.0074	0.36

Table 4. Comparison of KS test p-values for fitting parameters taken from [20]

stations. In *Proceedings of SIGMETRICS'95*, pages 267–278, 1995.

[4] R. H. Arpaci, A. C. Dusseau, A. M. Vahdat, L. T. Liu, T. E. Anderson, and D. A. Patterson. The Interaction of Parallel and Sequential Workloads on a Network of Workstations. In *Proceedings of the 1995 ACM SIGMETRICS Conference*, pages 267–278, May 1995.

[5] R. Bhagwan, S. Savage, and G. Voelker. Understanding Availability. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS 2003)*, 2003.

[6] G. Bosilca, A. Bouteiller, F. Cappello, S. Djilali, G. Fedak, C. Germain, T. Herault, P. Lemarinier, O. Lodygensky, F. Magniette, V. Neri, and A. Selikhov. Mpichv: Toward a scalable fault tolerant mpi for volatile nodes. In *Proceedings of SC'2002*. IEEE, Nov 2002.

[7] J. Chu, K. LAbonte, and B. Levine. Avaialbilie and Locality Measurements of Peer-to-Peer File Systems. In *Proceedings of ITCom: Scalability and Traffic Control in IP Networks (ITCom 2003)*, July 2003.

[8] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In *GGF*, 2002.

[9] L. Gao and G. Malewicz. On Scheduling Mesh-Structured Computation on Unreliable Computers on the Internet. In *Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS 2004)*, December 2004.

[10] C. Germain, V. Nri, G. Fedak, and F. Cappello. XtremWeb: Building an Experimental Platform for Global Computing. In *GRID 2000*, pages 91–101.

[11] G. Ghare and L. Leutenegger. Improving Speedup and Response Times by Replicating Parallel Programs on a SNOW. In *Proceedings of the 10th Workshop on Job Scheduling Strategies for Parallel Processing (WJSSPP'04)*, June 2004.

[12] D. Kondo. *Scheduling Task Parallel Applications on Enterprise Desktop Grids*. PhD thesis, Dept. of Computer Science and Engineering, University of California at San Diego, July 2005.

[13] D. Kondo, A. Chien, and H. Casanova. Resource Management for Short-Lived Applications on Enterprise Desktop Grids. In *Proceedings of the High Performance Computing, Networking and Storage Conference (SC2004)*, November 2004.

[14] D. Kondo, G. Fedak, F. Cappello, and H. Casanova. On Resource Volatility in Enterprise Desktop Grids. In *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing (e-Science 2006)*, December 2006.

[15] S. T. Leutenegger and X.-H. Sun. Distributed computing feasibility in a non-dedicated homogeneous distributed system. In *Proceedings of the 1993 ACM/IEEE Conference on Supercomputing (SC'03)*, pages 143–152, November 1993.

[16] M. Litzkow and M. Livny. Experiences with the Condor disributed batch system. In *IEEE Workshop on Experimental Distributed Systems*, pages 97–101, October 1990.

[17] D. Long, A. Muir, and R. Golding. A Longitudinal Survey of Internet Host Reliability. In *Proceedings of the 14th Symposium on Reliable Distributed Systems (SRDS'95)*, pages 2–9, 1995.

[18] G. Malewicz. Parallel scheduling of complex dags under uncertainty. In *Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'05)*, pages 66–75, 2005.

[19] G. Malewicz, I. Foster, A. Rosenberg, and M. Wilde. A Tool for Prioritizing DAGMan Jobs and Its Evaluation. In *Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing (HPDC-15)*, June 2006.

[20] D. Nurmi, J. Brevik, and R. Wolski. Modeling Machine Availability in Enterprise and Wide-area Distributed Computing Environments. August 2005.

[21] V. Ricci. Fitting distributions with r. `http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf`, Feb 1995.

[22] L. F. G. Sarmenta. Volunteer Computing. In *Proceedings of the 11th Euro-Par Conference (Euro-Par 2005)*, March 2001.

[23] S. Saroiu, p. Gummadi, and S. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of the Multimedia Computing and Networking Conference (MMCN'02)*, January 2002.

[24] M. Snir, S. W. Otto, S. Huss-Lederman, D. W. Walker, and J. Dongarra. *MPI the complete reference*. The MIT Press, 1996.

[25] Sullivan, Werthimer, Bowyer, Cobb, Gedye, and Anderson. A New Major SETI Project based on project SERENDIP data and 100,000 Personal Computers. In *Astronomical and Biochemical Origins and the Search for Life in the Universe*, 1997.

[26] D. Thain, T. Tannenbaum, and M. Livny. Distributed Computing in Practice: The Condor Experience. *Concurrency and Computation: Practice and Experience*, 17(2-4):323–356, 2005.

[27] United Devices Inc. `http://www.ud.com/`.