An Analysis of Availability Distributions in Condor*

Rich Wolski, Daniel Nurmi, John Brevik
University of California, Santa Barbara

1 Introduction

Recent research results [4, 11, 12, 27, 9] and infrastructure efforts [25, 24, 36, 15, 1] demonstrate the potential effectiveness of large-scale distributed computing. However, effective scheduling in these environments depends, increasingly, on the ability to tolerate the dynamic performance response exhibited by the underlying resources. Our approach has been to use first modeling (to understand the dynamics) and then prediction (to allow schedulers to avoid unfavorable conditions) as a way of obtaining performance in these settings [6].

In this work, we investigate the dynamics exhibited by the production Condor [35] pool at the University of Wisconsin with the goal of understanding its distributional properties. Condor is a cycle-harvesting service originally designed to launch and control "guest" user jobs (in batch mode) on idle workstations. Since its inception in 1985, however, it has expanded to include the ability to run in dedicated mode on clusters, to "glide in" to systems that are not strictly dedicated to Condor, and to "flock" jobs from one site to another based on pre-determined Service Level Agreements (SLAs). Thus it has developed from an enterprise-wide desktop system into a full-fledged global computing infrastructure over its lifetime.

The Condor project maintains a large active Condor pool at the University of Wisconsin that comprises campus desktop machines, dedicated clusters, and an automatic migration capability (called flocking) that can farm jobs to the National Center for Supercomputer Applications (NCSA) and other sites. In what is termed the "vanilla universe," a job submitted to Condor is directed to an idle machine, whence the job's standard input and output are redirected back to the submission machine. When a user or non-Condor job becomes active on the resource (i.e. the machine is no longer available for an external Condor job) the Condor job using the machine is terminated.

During the 26-month period from April 1, 2003 through July 1, 2005 we deployed an availability sensor designed for the Network Weather Service (NWS) to sample the durations of availability that "standard" jobs in the vanilla universe experience. In this paper, we describe our efforts to model the distribution of availability which we measure as the duration of time between when Condor schedules a job to a machine and when the job is terminated due to eviction. The results we have obtained are part of a larger effort that includes new predictive capabilities [5], new scheduling capabilities [26] and efforts to translate machine availability to application-level performance measurements [17]. Taken together, this effort embodies a new and comprehensive approach to modeling Global computing systems.

*This work was sponsored, in part, by a grant from the National Science Foundation numbered NGS-0305390.

1-4244-0910-1/07/$20.00 ©2007 IEEE

2 Automatically Determining an Availability Distribution

We have gathered machine availability data from from the vanilla universe in the University of Wisconsin Condor pool using the *Network Weather Service* (NWS) [39, 40, 41, 27],a distributed, robust, and scalable performance monitoring and forecasting system developed to support Grid and global computing. After studying individual machine traces, we have found that the two distribution families that consistently fit the availability data we have gathered most accurately are the Weibull and the Log-normal. These results are somewhat surprising, since a variety of previous efforts have focused on either exponential [37, 19, 31, 32, 20, 33, 42] or Pareto [14, 30, 29, 38, 8, 18] models of behavior. We compare the effectiveness of these more traditional approaches to our findings in the next section.

The *Weibull distribution* is often used to model the lifetimes of objects, including physical system components [3, 22] and also to model computer resource availability distributions [16, 34]. Hyperexponentials have been used to model machine availability previously [23] and are especially useful for observed data requiring a model that can approximate a wide variety of different shapes. Following are the formulas for the statistical models we compare in this work, along with a description of how we automatically estimate the model parameters from given a sample data set. Thus, with this system, we extract historical availability information from the NWS and generate a statistical models of previous availability. We can then compare these models in terms of their accuracy and "fit."

2.1 Weibull Distribution

The density and distribution functions f_w and F_w respectively for a Weibull distribution are given by

$$f_w(x) = \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^\alpha} \quad (1)$$

$$F_w(x) = 1 - e^{-(x/\beta)^\alpha} \quad (2)$$

The parameter α is called the *shape* parameter, and β is called the *scale* parameter [1]. Note that the Weibull distribution reduces to an exponential distribution when $\alpha = 1$.

2.2 Hyperexponential Distribution

Hyperexponentials are distributions formed as the weighted sum of exponentials, each having a different parameter. The density function is given by

$$f_H(x) = \sum_{i=1}^{k} [p_i \cdot f_{e_i}(x)] \tag{3}$$

where

$$f_{e_i}(x) = \lambda_i e^{-\lambda_i x} \tag{4}$$

defines the density function for an exponential having parameter λ_i. In the definition of $f_H(x)$, all $\lambda_i \neq \lambda_j$ for $i \neq j$, and $\sum_{i=1}^{k} p_i = 1$. The distribution function is defined as

$$F_H(x) = 1 - \sum_{i=1}^{k} p_i \cdot e^{-\lambda_i x} \tag{5}$$

for the same definition of $f_{e_i}(x)$. Thus, to fit a hyperexponential to a given data set, the value of k, each λ_i and each p_i must be estimated.

2.3 Log-normal Distribution

A population has a Log-normal distribution if the logarithm of the values in question has a normal distribution The density and distribution functions $f_{ln}(x)$ and $F_{ln}(x)$ respectively are given by

$$f_{ln}(x) = (1/(x\sigma\sqrt{2\pi}))e^{-(ln(x)-\mu)^2/(2*\sigma^2)} \tag{6}$$

$$F_{ln}(x) = 1/2 * 1/2(erf((ln(x) - \mu)/(\sigma\sqrt{2}))) \tag{7}$$

where μ and σ are the mean and standard deviation of logarithm of the population values and *erf* is the "error function" or cumulative distribution function from a standard normal .

2.4 Distribution Parameter Estimation

For repeatability, we describe the exact method used to perform all of the model fitting in this work. Given a set of sample data $\{x_1 ... x_n\}$, there are many common techniques for estimating distribution parameters based on some set of sample data, including visual inspection (*e.g.* using a two-dimensional graph) and analytic methods. A commonly accepted approach to the general problem of parameter estimation is based on the principle of *maximum likelihood*. The maximum likelihood estimator (MLE) is calculated for any data set, based on the assumptions that each of the sample data points x_i is drawn from a random variable X_i an that the X_i are independent and identically distributed (i.i.d.). The method defines the *likelihood function* L, depending on the parameters of the distribution, as the product of the density function evaluated at the sample points. Thus in the case of the Weibull distribution, L will be a function of α and β given by

$$L(\alpha, \beta \mid \{x_i\}) = \prod_i f(x_i \mid \alpha, \beta) = \prod_i \alpha\beta^{-\alpha} {x_i}^{\alpha-1} e^{-(x/\beta)^\alpha} \tag{8}$$

Roughly speaking, maximizing L is equivalent to maximizing the joint probability that each random variable will take on the sample value. Large values of the density function correspond to data that is "more likely" to occur, so larger values of L correspond to values of the parameters for which the data was "more likely" to have been produced. Thus, the MLE for the parameters is simply the choice of parameters (if it exists) which maximizes L. Our approach to computing MLE parameters numerically is to set the partial derivatives of $\log -likelihood$ function equal to 0 and solve for the distribution parameters.

We have implemented a software system that takes a set of measurements as an ordinary text file and computes the MLE Weibull and Log-normal automatically (as well as a variety of other distributions). Perhaps unsurprisingly, the quality of the numerical methods that we use is critical to the success of the method. In particular, the MLE computations can involve hundreds or thousands of terms (the data sets can be quite large) and thus require robust and efficient techniques. At present, the implementation uses a combination of the Octave [28] numerical package, Mathematica [21] (for solver quality). The resulting system, however, takes data (as described in Section 3) and automatically determines the necessary parameters.

The case of the hyperexponential is somewhat different. For a specified value of k (which indicates how many phases will be included in the hyperexponential), one can in principle set up and solve the necessary optimization problem to find MLE values for the remaining $2k - 1$ parameters. However, this problem, even small values of k, is in general too complex for commonly available computers to solve, especially for the larger data sets. Therefore, we used the EMpht software package [10] for all estimated hyperexponentials in this paper. EMpht implements the *estimation-maximization* (EM) algorithm described in [2]. While this software was able to compute the EM estimate of the 2-phase hyperexponential parameters for the smaller data set described in Section 4, it is still too computationally intensive for us to use with the larger data sets. We are studying this issue as part of our future work.

The number of exponential phases (denoted by k) that make up a hyperexponential, on the other hand, is a parameter that must be specified rather than estimated. Our approach is is to use EMpht to estimate parameters and then to calculate the log-likelihood values produced by the data for successively

[1] The general Weibull density function has a third parameter for *location*, which we can eliminate from the density simply by subtracting the minimum lifetime from all measurements. In this paper, we will work with the two-parameter formulation.

larger values of k. The algorithm terminates when an additional phase produces no discernible improvement in the metrics.

3 Experimental Data

Condor [7, 35] is a cycle-harvesting system designed to support high-throughput computing. Under the Condor model, the owner of each machine allows Condor to launch an externally submitted job (*i.e.*, one not generated by the owner) when the machine becomes "idle." Each owner is expected to specify when his or her machine can be considered idle with respect to load average, memory occupancy, keyboard activity, etc. When Condor detects that a machine has become idle, it takes an unexecuted job from a queue it maintains and assigns it to the idle machine for execution. If the machine's owner begins using the machine again, Condor detects the local activity and evacuates the external job. The result is that resource owners maintain exclusive access to their own resources, and Condor uses them only when they would otherwise be idle.

In this study, we take advantage of the vanilla (*i.e.*, terminate-on-eviction) execution environment to build a Condor occupancy monitor. A set of monitor processes (10 in this study) are submitted to Condor for execution. When Condor assigns a process to a processor, the process wakes periodically and reports the number of seconds that have elapsed since it began executing. When that process is terminated (due to an eviction) the last recorded elapsed time value measures the occupancy the sensor enjoyed on the processor it was using. We associate availability with Internet address and port number; therefore, if a monitor process is subsequently restarted on a particular machine (because Condor determined the machine to be idle), the new measurements will be associated with the machine running the process. In our study, Condor used 900 different Linux workstations to run the monitor processes over the 26-month measurement period.

Measuring processor availability in this way introduces "load" into Condor system in the form of sensor processes that use their occupancy time simply to record their occupancy time. Condor uses a sophisticated (and unrevealed) scheduling mechanism to decide when processes should run, and on what machine. It is possible that by introducing sensor load our measurements are preventing "real" work from being done, and such a perturbation affects the idle-busy cycle of each machine. Note that the name of the binary launched by Condor is obfuscated, however, so a machine owner cannot make a reclamation decision based on the name of the Condor processes running on his or her machine. That is, the machine owners at the University of Wisconsin could not tell the difference between our sensors and other computational work Condor assigned to the various machines. Notice also that in this study we consider only the availability of each machine to a Condor user once the machine is assigned to a process running on behalf of that user. We do not consider the time between assignments during which a particular machine is either busy because its owner is using it, or because Condor as scheduled other useful work.

4 Results

We are interested in the utility of different automatically fit models in describing the Condor data. We begin with a graphical analysis in which we compare the fits of an MLE Weibull having parameters $\alpha = 0.46$ and $\beta = 6227$ and a Log-normal distribution in which $\mu = 7.6$ and $\sigma = 2.2$ to the empirical data in cumulative distribution form (CDF) (Figure 1). To better show any differences in convexity, we put the x-axis on a log scale. In the figure, the empirical CDF includes all

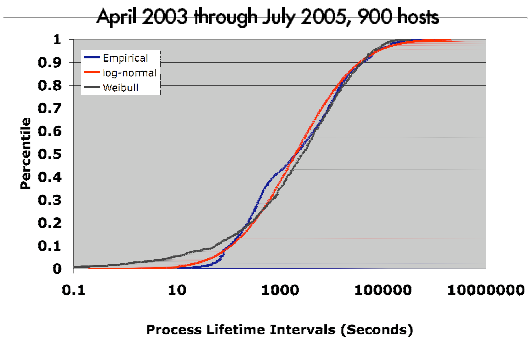

Figure 1: Comparison of Weibull and Log-normal Fits to Empirical Availability Data.

availability measurements taken from all 900 hosts during the observation period. The Weibull curve does not fit the tails of the distribution as well as the Log-normal does as evidenced by the greater density near the left tail of its curve in the figure. In contrast, the Log-normal fit is quite good throughout. Notice, however, that the empirical curve seems to have a pair of inflection points near its center that neither method captures. It may be that a hyperexponential distribution will be able to capture this feature better than either Weibull or Log-normal distributions. the computational complexity associated with automatically fitting even a 2-phase hyperexponential to this data makes it a subject of our future work.

Thus, it is clear from the figure that the two-parameter Log-normal distribution is a good candidate for modeling availability in the general Condor pool. A second question we wished to investigate is whether the distributional characteristics Condor were the for desktop machines as they are for the dedicated clusters. Recall that the Wisconsin Condor pool includes both cycle-harvested desktop resources and clusters that serve as dedicated Condor "compute engines."

While we do not possess detailed information on the actual local administration policies governing each machine, by examining the Domain Name Services (DNS) names associated

with each IP address in the study, we were able to perform a rough classification. In so doing, we divide the data set into one containing availability

In Figure 2 we compare the MLE Log-normal to the empirical CDF for only those hosts whose DNS name seemed to indicate that they were part of a cluster. The parameters for

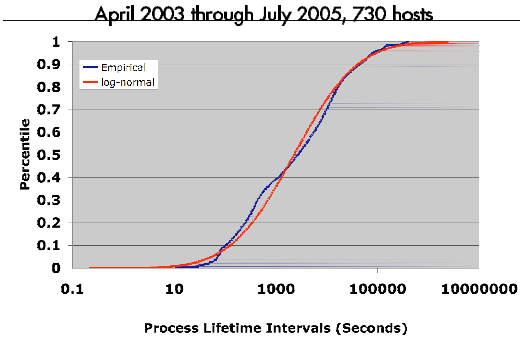

Figure 2: Comparison of Log-normal Fit to Empirical Availability Data for Hosts with Cluster-like DNS Names.

the MLE Log-normal shown in this figure are $\mu = 7.7$ and $\sigma = 2.2$. Clearly, the shape of the availability curve is is predominantly determined by machines that are part of Condor clusters and not desktops. For the cluster systems, occupancy is determined by competitive load and by the priority scheduling mechanisms that Condor uses internally as well as external load introduced by local cluster users. From Figure 2 the distribution of availability durations that was available to our sensor is well-modeled by a Log-normal distribution.

This observation indicates that the Condor scheduling mechanisms are effective in delivering reasonable availability periods to user applications. It has been well-documented that process lifetimes are well-modeled by heavy-tailed distributions [13]. Despite the potential for large variance in lifetime which might, in turn, impose a large variance in availability, the actual observed availability durations are not heavy-tailed due to Condor's scheduling algorithms.

In Figure 3 we show the empirical CDF for the availability durations that we observed from machines in the Condor pool with DNS names that did not imply membership in a cluster. Note that in the case of cluster processors, this categorization methodology is most likely conservative; the DNS names are fairly obviously part of a cluster. The complement, however, is less certain as it is possible that cluster machines may have been identified by only their head node, and the head node did not have an obvious cluster-like name. The curves in this figure are difficult to differentiate without the use of color. In print form, moving from left to right the order in which each curve touches the x-axis is Weibull, 2-phase hyperexponential, Log-normal and empirical. The parameters for the MLE Weibull, in this case, are $\alpha = 0.56$ and $\beta = 975$ and for

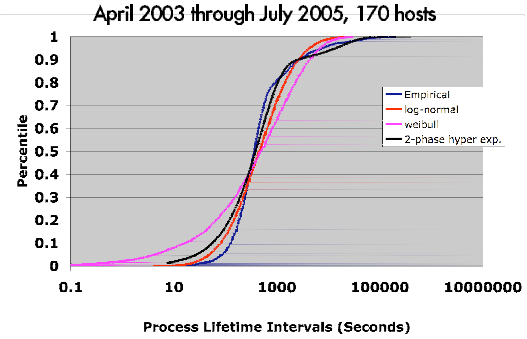

Figure 3: Comparison of Various Fits to Empirical Availability Data for Hosts without Cluster-like DNS Names.

the MLE Log-normal are $\mu = 6.1$ and $\sigma = 1.35$. In this case, because the dataset is smaller, we were also able to use the EM algorithm to fit a hyperexponential with parameters $p_0 = 0.88$, $\lambda_0 = -0.002$, and $\lambda_1 = -0.00005$.

From the figure, it is not clear which method fits "best." The Weibull distribution overestimates the probability density at the left end by substantial amount. The empirical data, itself, shows a probability density that is more concentrated near the center of the curve than any of the models are able to capture. To the left of the mode, the Log-normal is closer, but to the right of it, the 2-phase hyperexponential is closest.

5 Discussion

In terms of model fitting, the strength of these results are mixed. For dedicated Condor clusters, the quality of the fit provided by an MLE determined Log-normal appears quite good. In a simulation context, for example, this correspondence is probably good enough to warrant its use. This circumstance is particularly fortuitous since the Log-normal is relatively easy to implement and cheap to compute. This result is especially important since no existing simulation mechanisms of which we are aware currently hypothesize or advocate the use of the Log-normal for Global computing systems such as Condor.

For desktop machines, however, the results are less clear. Certainly, the availability durations are far from heavy-tailed as the Weibull (the distribution among those we tested with the greatest right-tail weight) seems to fit the least well. In this case, the decision on which distribution to use may depend on whether it is more important to be "close" on the left side of the mode or on the right. That is, if one were interested in in a small quantile, then an MLE Log-normal would most likely to yield a good result. Alternatively, if a large quantile is the statistic of interest, an EM-determined 2-phase hyperexponential might be more appropriate. Again, however, we know of no simulation environment in which these results

have been actualized.

6 Conclusion

The focus of this effort is on an examination of which automatically determined models best capture the distributional properties of Condor – an active and successful Global computing system. To facilitate this study, we have gathered an extensive survey of Condor availability using a new NWS sensing capability. We present methodologies for automatically fitting parametric models to machine availability data. We find that the Log-normal model most suitable for part of the Condor survey (those machines that appear to be part of a Condor-dedicated cluster) but that for desktop machines, no single model seems definitively best.

Taken as part of a larger effort [5, 17, 26] this work constitutes an important step toward achieving a new and powerful global computing infrastructure. Through a combination of newly developed modeling and prediction techniques, their application in simulation to the problem of scheduling, and their empirical verification with simulation and functioning application, our goal is to lay the groundwork for the scientific study of next generation distributed computing.

References

[1] The Asia Pacific Grid Home Page. `http://www.apgrid.org`.

[2] S. Asmussen, O. Nerman, and M. Olsson. Fitting phase-type distributions via the em algorithm. *Scandinavian Journal of Statistics*, 23:419–441, 1996.

[3] C. E. Beldica, H. H. Hilton, and R. L. Hinrichsen. Viscoelastic beam damping and piezoelectric control of deformations, probabalistic failures and survival times.

[4] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, L. J. Dennis Gannon, K. Kennedy, C. Kesselman, D. Reed, L. Torczon, , and R. Wolski. The GrADS project: Software support for high-level grid application development. *International Journal of High-performance Computing Applications*, 15(4):327–344, Winter 2001.

[5] J. Brevik, D. Nurmi, and R. Wolski. Quantifying machine availability in networked and desktop grid systems. In *Proceedings of CCGrid04*, April 2004.

[6] W. Chrabakh and R. Wolski. GrADSAT: A Parallel SAT Solver for the Grid. In *Proceedings of IEEE SC03*, November 2003.

[7] Condor home page – `http://www.cs.wisc.edu/condor/`.

[8] M. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5, December 1997.

[9] The Data Grid Project. `www.eu-datagrid.org`.

[10] Empht home page. Available on the World-Wide-Web. `http://www.maths.lth.se/matstat/staff/asmus/pspapers.html`.

[11] GrADS. `http://hipersoft.cs.rice.edu/grads`.

[12] The Grid Physics Network Home Page. `http://www.griphyn.org`.

[13] M. Harchol-Balter and A. Downey. Exploiting process lifetime distributions for dynamic load balancing. In *Proceedings of the 1996 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, 1996.

[14] M. Harcol-Balter and A. Downey. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems*, 15(3):253–285, 1997.

[15] The NASA Information Power Grid Home Page. `http://www.ipg.nasa.gov`.

[16] X. J., K. Z., and I. R. K. Networked windows nt system field failure data analysis.

[17] D. Kondo, H. Casanova, E. Wing, and F. Berman. Models and Scheduling Mechanisms for Global Computing Applications. In *Proc. of the International Parallel and Distributed Processing Symposium (IPDPS), Fort Lauderdale, Florida*, April 2002.

[18] W. Leland and T. Ott. Load-balancing heuristics and process behavior. In *Proceedings of Joint International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS '86)*, pages 54–69, May 1986.

[19] D. Long, A. Muir, and R. Golding. A longitudinal survey of internet host reliability. In *14th Symposium on Reliable Distributed Systems*, pages 2–9, September 1995.

[20] D. D. E. Long, J. L. Carroll, and C. J. Park. A study of the reliability of internet sites. In *Proceedings of the 10th IEEE Symposium on Reliable Distributed Systems (SRDS91)*, 1991.

[21] Mathematica by Wolfram Research. `http://www.wolfram.com`.

[22] S. M.H., R. M.L., B. M.C., S. P., , and B. S. Mechanical properties of fabrics woven from yarns produced by different spinning technologies: Yarn failure in fabric.

[23] M. Mutka and M. Livny. Profiling workstations' available capacity for remote execution. In *Proceedings of Performance '87: Computer Performance Modelling, Measurement, and Evaluation, 12th IFIP WG 7.3 International Symposium*, December 1987.

[24] The National Computational Science Alliance. `http://www.ncsa.uiuc.edu`.

[25] The National Partnership for Avanced Computational Infrastructure. `http://www.npaci.edu`.

[26] D. Nurmi, R. Wolski, and J. Brevik. Model-based checkpoint scheduling for volatile resource environments. Technical Report CS2004-25, U.C. Santa Barbara Computer Science Department, November 2004.

[27] The network weather service home page – `http://nws.cs.ucsb.edu`.

[28] GNU Octave Home Page. `http://www.octave.org`.

[29] V. Paxon and S. Floyd. Wide area traffic: the failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3), 1995.

[30] V. Paxon and S. Floyd. Why we don't know how to simulate the internet. In *Proceedings of the Winder Communication Conference also `citeseer.nj.nec.com/paxon97why.html`*, December 1997.

[31] J. Plank and W. Elwasif. Experimental assessment of workstation failures and their impact on checkpointing systems. In *28th International Symposium on Fault-Tolerant Computing*, pages 48–57, June 1998.

[32] J. Plank and M. Thomason. Processor allocation and checkpoint interval selection in cluster computing systems. *Journal of Parallel and Distributed Computing*, 61(11):1570–1590, November 2001.

[33] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and K. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *In Proc. SIGCOMM (2001)*, 2001.
[34] H. T., M. P. M., and N. T. D. The shape of failure.
[35] T. Tannenbaum and M. Litzkow. The condor distributed processing system. *Dr. Dobbs Journal*, February 1995.
[36] The TeraGrid Home Page. `http://www.teragrid.org`.
[37] N. Vaidya. Impact of checkpoint latency on overhead ratio of a checkpointing scheme. *IEEE Transactions on Computers*, 46(8):942–947, August 1997.
[38] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level. In *SIGCOMM'95 Conference on Communication Architectures, Protocols, and Applications*, pages 110–113, 1995.
[39] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. *ACM SIGMETRICS Performance Evaluation Review*, 30(4):41–49, March 2003.
[40] R. Wolski, G. Obertelli, M. Allen, D. Nurmi, and J. Brevik. Predicting grid resource performance on-line. In A. Zomaya, editor, *Handbook of Innovative Computing: Models*. Springer-Verlag, Fall 2004.
[41] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5-6):757–768, October 1999.
[42] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. A resilient global-scale overlay for service deployment. *(to appear) IEEE Journal on Selected Areas in Communications*.