Models and Modeling Infrastructures for Global Computational Platforms*

Rich Wolski, Daniel Nurmi, John Brevik
University of California, Santa Barbara
Henri Casanova, Andrew Chien
University of California, San Diego

1 Introduction

Recent research results [8, 26, 27, 47, 19] and infrastructure efforts [45, 44, 62, 30, 3] demonstrate the potential effectiveness of large-scale distributed computing. Effective scheduling based on empirically verifiable models has emerged as a key factor in these successes. Moving to a new truly global computing capability will similarly depend critically on new models and scheduling techniques. The development and instantiation of models for computing platforms are critical first steps for studying the feasibility and scalability of applications, as well as for developing methodologies to enhance application performance. Our focus, in this proposal, is on *the modeling technologies that will enable performance-engineering of such global computations*. Specifically, these new modeling technologies will support resource characterization and scheduling techniques thereby enabling robust, predictable performance.

The increasing attractiveness of global computing approaches is fueled by dramatic advances in networking technology, and the penetration of PCs into the home. Numerous projects (*e.g.* SETI@home, Great Internet Mersenne Prime Search, and Compute Against Cancer) provide a striking demonstration of the power available from networked PCs world-wide. To date, many academic and commercial projects are attempting to harness these resources for compute intensive applications [56, 10, 22, 34, 13, 21] and for developing distributed storage systems [33, 17, 25, 54, 18], either world-wide or within institutions in an "enterprise" fashion. We use the term *global computing* for these projects, as they can exploit a potentially world-wide universe of resources. The key differentiating characteristics of these systems is that application performance must be amalgamated from:

- desktop / laptop PC resources which are unreliable and prone to reclamation by users,
- intermittent and highly fluctuating network connectivity,
- resources that span organizational boundaries each having unique (potentially fluctuating) security and access control policies, and
- an application structure that permits flexible concurrency management and scheduling.

Early adopters of this emerging paradigm have used it to deliver massive compute capacity to their scientific, engineering, or business applications [2, 6, 38, 1, 43, 58]. Such applications often involve large numbers of independent (or nearly so) computation and data objects, and are tolerant of high network latencies and resource failures. These examples point to the dramatic advances that may be achieved if three orders of magnitude more compute power than is available from single-site resources is culled from a global resource pool. Applications already moving to this new level of capability include GIMPS [24], and Folding@Home [5].

These early examples demonstrate the capabilities of global computing to solve real problems, but their customized approaches have yet to be generalized. In particular, no well-defined method for modeling resource usage by applications that can take advantage of massively heterogeneous, volatile, unreliable resources for performance yet exists. As a result, achieving robust, predictable performance poses a significant challenge. Our work will address this deficiency directly by providing new modeling capabilities that support performance scheduling in global computing environments.

At the same time, current Grid computing platforms are more tightly controlled than the nascent global computing platforms that are available. As a result, resource availability tends to be more predictable since the administrative policies are in place to ensure more availability and less volatility. The trend in Grid systems, however, is toward dynamically coordinated Web services [23] that are compatible with W3C standards. A lack of effective unifying models and scheduling techniques poses a serious impediment to this continued convergence between the Grid computing approach and the Internet computing approach. In particular, platform resource models that capture the performance behavior of *both* Grid and global computing resources are key to the next generation of effective distributed high-performance applications.

A widely applicable global compute capability can only be realized if models are developed and instantiated that allow

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

the engineering of appropriate scheduling and deployment of applications, and for predicting their performance. Such models must characterize resource capabilities, dynamic behavior, availability, failure, and connectivity. Due to lack of such models, current global computing applications are customized point-solutions (*e.g.* high throughput with a centralized server) with unpredictable performance characteristics (*e.g.* availability and response time).

Platform and resource modeling has been an important focus in the Grid computing community and has led to significant number of successful applications [55, 51, 42, 57]. However, in the face of the current evolutions, it is clear that new models and modeling techniques are required both for effective global computing and for next generation Grid computing. In particular, the uncoordinated burstiness of resource availability, caused by user reclamation of resources and intermittent network connectivity, is orders of magnitude larger than that of previously studied distributed computing platforms. New measurement, analysis, prediction, and simulation techniques must be investigated. Only with these models will it be possible to enable a wide range of applications to compute effectively at a global level.

Our approach to achieving the modeling and simulation capabilities that are necessary to enable global computing comprise three objectives proposed herein. We must

1. *Develop novel models of complex emerging global and Grid computing platforms. These models will target resource availability as well as resource classification, and will attempt to unify the performance characteristics of both platform types.*

2. *Use these models to develop resource classification and scheduling strategies which enable robust, predictable performance and deliver high availability for applications*

3. *We will validate our approach with two "real-world" global computing applications, using both novel simulation techniques and, if possible, a real global computing system.*

One of our goals is to develop models of resource availability as they are fundamental to an understanding of overall platform behavior and scheduling policies. Our previous modeling and scheduling efforts [70, 12, 67, 68, 57, 9] capture and make effective use of predicted performance levels, but do not include predictions of resource loss or failure. For initial Grid efforts, resource availability predictions did not prove critical. Moving to a more global computing context, however, will require availability and failure predictions that can be used as the basis for new performance-oriented scheduling techniques.

In addition, we believe that the issue of *resource classification* continues to be fundamental. One of the major challenges for global computing is is the ability to impose a virtual structure (possibly application-specific) over the potentially useful resources at any given time making it possible for schedulers to reason about platform utilization and performance. We must be able to cluster resources according to performance response, thereby defining resource categories (*e.g.* common availability behaviors, coarse notions of network proximity).

In this paper, we briefly describe our efforts to automatically determine good parametric models of machine availability. The results we have obtained are part of a larger effort that includes new predictive capabilities [11], new scheduling capabilities [46] and efforts to translate machine availability to application-level performance measurements [32]. Taken together, this effort embodies a new and comprehensive approach to modeling Global computing systems.

2 Automatically Determining an Availability Distribution

We have gathered machine availability data from a variety of desktop, Internet, and cycle-harvesting systems using the Network Weather Service (NWS) [66, 69, 70, 47] – distributed, robust, and scalable performance monitoring and forecasting system developed to support Grid and global computing. After studying individual machine traces (some spanning almost two years) we have found that the two distribution families that consistently fit the data we have gathered most accurately are the Weibull and the hyperexponential. These results are somewhat surprising, since a variety of previous efforts have focused on either exponential [64, 36, 52, 53, 37, 59, 71, 72] or Pareto [29, 50, 49, 65, 15, 35] 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 [7, 40] and also to model computer resource availability distributions [31, 60]. Hyperexponentials have been used to model machine availability previously [41] especially when observed data requires a model which can approximate a wide variety of shapes. Following are the equations for the 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. For a specified value of k (which indicates how many phases will be included in the hyperexponential), an MLE technique can be used to determine the remaining $2k - 1$ parameters. However, the optimization problem that arises for even small values of k is often too complex for commonly available computers to solve, especially for larger data sets [2]. As a result, we used the EMpht software package [20] in place of an MLE procedure for all estimated hyperexponentials in this paper. EMpht implements the estimation maximization (EM) algorithm described in [4].

The number of exponential phases (denoted by k) that make up a hyperexponential, however, is a free parameter that must be specified rather than estimated. Our approach is is to use EMpht to estimate parameters for successively larger values of k and then to calculate goodness-of-fit metrics for each. The algorithm terminates when an additional phase produces no discernible improvement in the metrics.

2.3 Exponential and Pareto Distributions

The probability density functions (denoted using lower-case f with a subscript) and distribution functions (upper-case F with a subscript) for the exponential and Pareto distributions are as follows:

$$f_e(x) = \lambda e^{-\lambda x} \tag{6}$$

$$F_e(x) = 1 - e^{-\lambda x} \tag{7}$$

$$f_p(x) = \frac{\alpha \beta^{\alpha}}{x^{\alpha+1}} \tag{8}$$

$$F_p(x) = 1 - \left(\frac{\beta}{x}\right)^{\alpha} \tag{9}$$

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

$$L(\alpha, \beta \,|\, \{x_i\}) = \prod_i f(x_i \,|\, \alpha, \beta) = \prod_i \alpha \beta^{-\alpha} {x_i}^{\alpha-1} e^{-(x/\beta)^{\alpha}} \tag{10}$$

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, Pareto, exponential and the EM-based hyperexponential automatically. 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 [48] numerical package, Mathematica [39] (for solver quality), and the above-mentioned EMpht. The resulting system, however, takes data (as described in Section 3) and automatically determines the necessary parameters.

[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.

[2] While we were able to make MLE estimates for Weibull and Pareto distributions for all data sets, the same numerical algorithms failed for all hyperexponential estimations.

3 Experimental Data

The data we use in this study measures resource availability in three different settings: a general student workstation laboratory at the University of California, Santa Barbara, the Condor [61] cycle-harvesting system at the University of Wisconsin, and on the Internet circa 1995 [36, 52]. These three data sets were obtained using different measurements of availability; our goal being to determine how sensitive our models are to the way in which availability is measured.

3.1 The UCSB CSIL Data Set

At UCSB, the computer science students are given unrestricted access to workstations located in several rooms on campus. Together, these systems make up the Computer Science Instructional Laboratory (CSIL). Physical access to the CSIL is provided to some (but not all) students 24 hours a day when school is in session, and via remote access at all other times to all computer science students. There are no administrator-scheduled reboots when school is in session; however, software failures, security breaches, and hardware failures result in unplanned restarts by the administrative staff. An interesting behavior we have noticed on these systems is that it appears students periodically "clean off" the machines by simply hitting the power switch and rebooting the systems. As anarchic as it may seem, we believe this mode of usage and administration is relatively common in today's publicly accessible computer workstation environments.

All CSIL workstations currently run Linux which records the time since reboot in the */proc* directory. We recorded the availability times from 83 of the CSIL workstations and recorded the duration between reboots during April and May of 2003, which corresponds to the bulk of the spring quarter. Thus the resultant data set captures a "production" use period for the CSIL machines and does not span a quarter break, during which a correlated reboot (for quarterly maintenance) is likely.

3.2 The Condor Data Set

Condor [14, 61] 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 210 different Linux workstations to run the monitor processes over the six-week measurement period.

3.3 The Long-Muir-Golding Data Set

In [36] the authors identify 1170 hosts connected to the Internet in 1995 that would cooperatively respond to a vacuous query of the *rpc.statd*; a Unix service which is commonly found on systems using the Network File System (NFS). The hosts were chosen to act as a "cross-section" of machines connected to the Internet, and a probing mechanism based on periodic but randomized RPC calls to *rpc.statd*. A successful response to an RPC constitutes a "heartbeat" for the machine in question, and failure to respond indicates machine failure. Long, Muir, and Golding use this data to make a convincing argument that availability is not accurately modeled by a Poisson process. More recently, Plank and Elwasif [52] and, separately, Plank and Thomason [53], have analyzed it extensively in terms of the suitability of Poisson and exponential models in the context of process checkpoint scheduling. In all three studies, the authors reach the same conclusion, which is that the models under study do not accurately reflect the behavior captured by the measurements.

4 Results

The goal of our study is to determine the value of using Weibull and hyperexponential distributions to model resource availability in contrast with previous approaches. Our method is to compare the MLE-determined Weibull and EMpht-determined hyperexponential to the MLE exponential and Pareto for each of the data sets discussed in the previous section. For reference, we have included the MLE-determined and EMpht-determined model parameters that were used for all fitted distributions discussed and shown in this work (Table 1). As we noted in the introduction, both exponential and the Pareto models have been used extensively to model resource and process lifetime. Thus the value we perceive is the degree to which the Weibull and hyperexponential models more accurately fit each data set.

Data Set	Weibull		Hyperexponential						Exp.	Pareto	
	α	β	p_1	p_2	p_3	λ_1	λ_2	λ_3	λ	α	β
CSIL	.545	275599	.464	.197	.389	$1 * 10^{-6}$	$2 * 10^{-4}$	$8 * 10^{-6}$	$2 * 10^{6}$	.087	1
Condor	.49	2403	.592	.408	NA	$3 * 10^{-3}$	$7 * 10^{-5}$	NA	.00018	.149	1.005
Long	.61	834571	.282	.271	.474	$3 * 10^{-7}$	$1 * 10^{-5}$	$1 * 10^{-6}$	$7 * 10^{7}$	.079	1

Table 1: Table of fitted model parameters

In each case, we use three different techniques to evaluate model fit: graphical; the Kolmogorov-Smirnov [16] (KS) test; and the Anderson-Darling [16] (AD) test. Graphical evaluation is often the most compelling method [63] but it does not provide the security of a quantified result. The other two tests come under the general heading of "goodness-of-fit" tests. Note that the best known goodness-of-fit test is based on the chi-squared distribution. However, this test is designed for use with categorical data, and therefore to use chi-squared with quantitative data requires that the data be artifically "binned" into discrete categories. For this reason, both the Kolmogorov-Smirnov and the Anderson-Darling tests are thought to be more appropriate for continuous distributions than the chi-squared test. We therefore use these methods in place of the more familiar one.

4.1 Graphical Analysis of The Availability Measurements

To gauge the fit of a specific model distribution to a particular data set, we plot the cumulative distribution function (CDF) for the distribution and the empirical cumulative distribution for the data set. The form of the CDF for the Weibull, hyperexponential, exponential and Pareto are given by equations 2, 5, 7, and 9 respectively (*cf.* Section 2). The empirical distribution function (EDF) is the CDF of the actual data; it is calculated by ordering the observed values as $X_1 < X_2 < \cdots < X_n$ and defining

$$F_e(x) = \begin{cases} 0, & x < X_1; \\ j/n, & X_j \leq x < X_{(j+1)}; \\ 1, & x \geq X_n. \end{cases} \qquad (11)$$

We start by comparing the empirical observations from the CSIL data set (as an EDF) to the CDF determined by the EMpht-estimated hyperexponential, and the MLE-estimated Weibull, exponential, and Pareto distributions. In all of the figures depicting distributions in this paper, the units associated with the x-axis are seconds of machine availability. We use a log scale for the x-axis to better expose the nature of each fit. Both the hyperexponential and the Weibull fit the data substantially better than either an exponential or Pareto; the hyperexponential is also able to capture the slight inflection around 10,000 seconds. As was previously mentioned, the choice of number of phases is a value specified by the user when attempting to fit a hyperexponential using the EMpht software. To determine the number of phases to report in the visual analysis, we start with a 2-phase hyperexponential, test the resulting fit with a Kolmogorov-Smirnov test and then repeat with an increased number of phases until the KS test result shows no improvement. In this case, for the CSIL data, the algorithm terminated using three phases.

For the Condor data set, the comparison (shown in Figures 5, 4, and 6) is more striking. Again, the hyperexponential (a 2-phase, in this case) appears to fit the shape of the curve most closely, and the Weibull appears a better choice than either exponential or Pareto. Note in particular how again the hyperexponential is able to capture the inflection points of the Condor EDF around 1000 seconds, while the Weibull is unable to do so.

Finally, the fits (3-phase hyperexponential in this case) for the Long, Muir, and Golding data are shown in Figures 8, 7, and 9.

The comparison is similar to that for the CSIL data. The multi-phase hyperexponential fits slightly better than a Weibull, and both are substantially better than an exponential or Pareto.

Of particular interest are the way in which each hypothetical distribution appears to match the tail of an EDF. In many application contexts, "tail behavior" can be important, especially if the presence or absence of rare occurrences must be modeled accurately. For example, previous research [28, 29] reveals Unix process lifetimes to be "heavy-tailed" and well-modeled by a Pareto distribution. Thus schedulers and process management systems must be designed for infrequently occurring processes that have very long execution times.

According to Figures 3, 6, and 9, however, a Pareto distribution would over-estimate the probability of very long-lived resources by a considerable amount. Indeed, it may be that while Unix process lifetime distributions are heavy tailed, if they are executed in distributed or global computing environments, many of them will be terminated by resource failure since the resource lifetime distributions (both EDFs and their matching Weibull and hyperexponential fits) have considerably less tail weight.

Even beyond the differences in the tails, however, we can clearly see that the general shape of the exponential and Pareto distributions do not seem to fit the sample CDFs well.

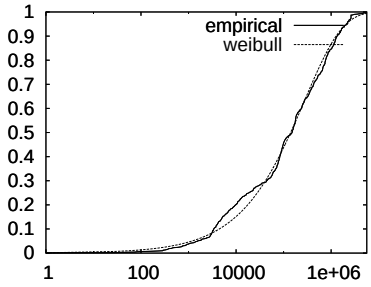

Figure 1: CSIL data with Weibull fit

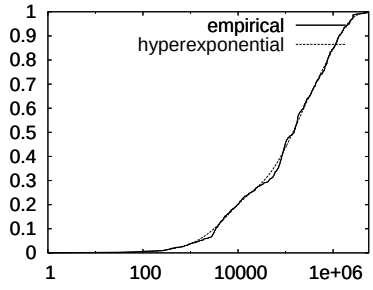

Figure 2: CSIL data with hyperexponential fit

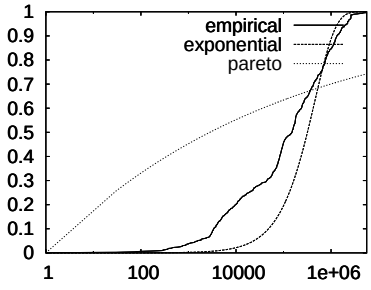

Figure 3: CSIL data with exponential and Pareto fits

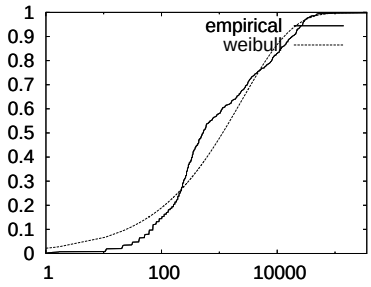

Figure 4: Condor data with Weibull fit

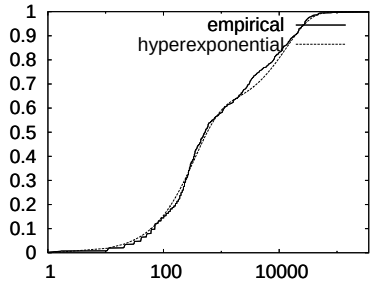

Figure 5: Condor data with hyperexponential fit

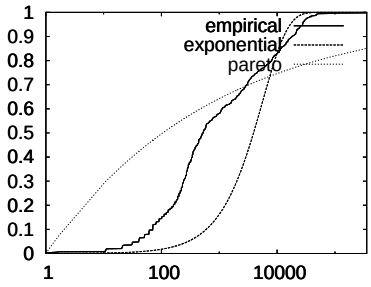

Figure 6: Condor data with exponential and Pareto fits

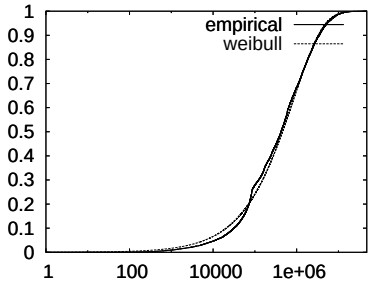

Figure 7: Long data with Weibull fit

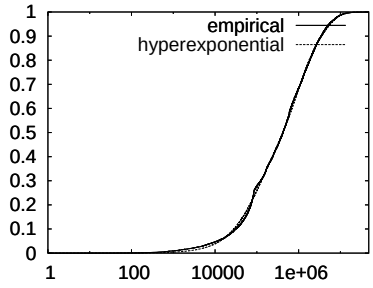

Figure 8: Long data with hyperexponential fit

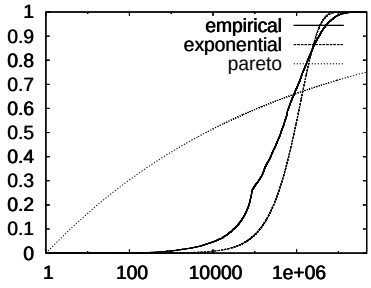

Figure 9: Long data with exponential and Pareto fits

5 Conclusion

The focus of this effort is on modeling and scheduling methodologies that will enable global computing, thus extending Computational Grids and high-performance distributed computing to a global level. In this vein, we present a methodology for automatically fitting parametric models to machine availability data. We find that Weibull and Hyper-exponential models are more suitable than other, more extensively studied alternatives such as exponential and Pareto models.

Taken as part of a larger effort [11, 32, 46] this work constitutes an important step toward achieving a new and powerful global computing infrastructure. Through a rigorous 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] D. Altilar and Y. Paker. An Optimal Scheduling Algorithm for Parallel Video Processing. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, 1998.

[2] S. Altschul, T. Madden, A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Res.*, 25:3389–3402, 1997.

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

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

[5] F. at home. `http://folding.stanford.edu`, 2001.

[6] J. Basney, M. Livny, and P. Mazzanti. Harnessing the Capacity of Computational Grids for High Energy Physics. In *Proceedings of the Conference on Computing in High Energy and Nuclear Physics*, 2000.

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

[8] 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.

[9] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application level scheduling on distributed heterogeneous networks. In *Proceedings of Supercomputing 1996*, 1996.

[10] The berkeley open infrastructure for network computing (boinc). `http://www.isi.edu/nsnam/ns`, 2001.

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

[12] H. Casanova, G. Obertelli, F. Berman, and R. Wolski. The AppLeS Parameter Sweep Template: User-Level Middleware for the +Grid. In *Proceedings of IEEE SC'00 Conference on High-performance Computing*, Nov. 2000.

[13] A. Chien, B. Calder, S. Elbert, and K. Bhatia. Entropia: Architecture and performance of an enterprise desktop grid system. *Journal of Parallel and Distributed Computing*, 2003.

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

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

[16] R. B. D'Agostino and M. A. Stephens. *Goodness-Of-Fit Techniques*. Marcel Dekker Inc., 1986.

[17] A. Demers, K. Petersen, M. Spreitzer, D. Terry, and M. Theimer. The Bayou architecture: Support for data sharing amon mobile users. In *Proceedings of the IEEE Workshop on Mobile Computing Systems & Applications*, Dec. 1994.

[18] edonkey. `http://www.edonkey2000.com/`, 2002.

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

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

[21] The Entropia Home Page. `http://www.entropia.com`.

[22] G. Fedak, C. Germain, V. N?ri, and F. Cappello. XtremWeb : A Generic Global Computing System. In *Proceedings of the Workshop on Global Computing on Personal Devices*, May 2001.

[23] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. January, 2002.

[24] T. G. I. M. P. S. (GIMPS). `http://www.mersenne.org/`, 2001.

[25] Gnutella. `http://www.gnutellanews.com`, 2001.

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

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

[28] 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.

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

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

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

[32] 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.

[33] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, 2000.

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

[35] 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.

[36] 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.

[37] 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.

[38] A. Majumdar. Parallel Performance Study of Monte-Carlo Photon Transport Code on Shared-, Distributed-, and Distributed-Shared-Memory Architectures. In *Proceedings of the 14th Parallel and Distributed Processing Symposium, IPDPS'00*, pages 93–99, May 2000.

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

[40] 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.

[41] 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.

[42] A. Narajan, M. Crowley, N. Wilkins-Diehr, M. Humphrey, A. Fox, A. Grimshaw, and C. Brooks. Studying protein folding on the grid: Experiences using charmm on npaci resources under legion. In *Proceedings 10th IEEE Symp. on High Performance Distributed Computing*, August 2002.

[43] A. Natrajan, M. Crowley, N. Wilkins-Diehr, M. Humphrey, A. Fox, and A. Grimshaw. Studying Protein Folding on the Grid: Experiences using CHARM on NPACI Resources under Legion. In *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, 2001.

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

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

[46] 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.

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

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

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

[50] 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.

[51] A. Petitet, S. Blackford, J. Dongarra, B. Ellis, G. Fagg, K. Roche, and S. Vadhiyar. Numerical libraries and the grid. In *Proceedings of IEEE SC'01 Conference on High-performance Computing*, November 2001.

[52] 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.

[53] 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.

[54] T. F. N. Project.

[55] M. Ripeanu, A. Iamnitchi, and I. Foster. Cactus application: Performance predictions in a grid environment. In *proceedings of European Conference on Parallel Computing (EuroPar) 2001*, August 2001.

[56] SETI@home. `http://setiathome.ssl.berkeley.edu`, March 2001.

[57] N. Spring and R. Wolski. Application level scheduling: Gene sequence library comparison. In *Proceedings of ACM International Conference on Supercomputing 1998*, July 1998.

[58] J. Stiles, T. Bartol, E. Salpeter, and M. Salpeter. Monte Carlo simulation of neuromuscular transmitter release using MCell, a general simulator of cellular physiological processes. *Computational Neuroscience*, pages 279–284, 1998.

[59] 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.

[60] H. T., M. P. M., and N. T. D. The shape of failure.

[61] T. Tannenbaum and M. Litzkow. The condor distributed processing system. *Dr. Dobbs Journal*, February 1995.

[62] The TeraGrid Home Page. `http://www.teragrid.org`.

[63] E. Tufte. *The Visual Display of Quantitative Information, 2nd Ed.* Graphics Press, May 2001.

[64] N. Vaidya. Impact of checkpoint latency on overhead ratio of a checkpointing scheme. *IEEE Transactions on Computers*, 46(8):942–947, August 1997.

[65] 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.

[66] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. *ACM SIGMETRICS Performance Evaluation Review*, 30(4):41–49, March 2003.

[67] R. Wolski, J. Brevik, C. Krintz, G. Obertelli, N. Spring, and A. Su. Running everyware on the computational grid. In *Proceedings of IEEE SC'99 Conference on High-performance Computing*, April 1999.

[68] R. Wolski, J. Brevik, C. Krintz, G. Obertelli, N. Spring, and A. Su. Writing programs that run everyware on the computational grid. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1066–1080, 2001.

[69] 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.

[70] 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.

[71] 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*.

[72] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, U.C. Berkeley Computer Science Department, April 2001.