Statistical Properties of Host Load in a Distributed Environment
(extended abstract)

Peter A. Dinda David R. O'Hallaron
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
(pdinda,droh)@cs.cmu.edu

February 25, 1998

1 Introduction

The distributed computing environment the most of us have access to in our workplaces consists of a collection of loosely interconnected hosts running vendor operating systems. Tasks are initiated independently by users and are scheduled locally by a vendor supplied operating system; there is no global scheduler that controls access to the hosts. As users run their jobs the computational load on the individual hosts changes over time.

Deciding how to map computations to hosts in systems with dynamically changing loads (what we will call the *mapping problem*) is a basic problem that arises in a number of important contexts, such as dynamically load-balancing the tasks in a parallel program [11, 1, 12], and scheduling tasks in a distributed soft real-time system [5, 9, 10].

Host load can have a significant effect on running time. Thus, a good mapping decision will require some sort of forecast of the loads on the remote hosts for the duration of the computation. Good forecasts will require some understanding of the qualititative and quantitative properties of load on real systems. Unfortunately, to date there has been little work on characterizing the properties of load, and there are no available load traces for practitioners to study and learn from.

This paper is a first step to a better understanding the properties of load on real systems. We collected week-long, 1 Hz resolution traces of the Unix load average on 38 different machines that we classify as production and research cluster machines, compute servers, or desktop workstations. Here, we present a detailed analysis of the statistical properties of those traces, and we contemplate the implications of those properties for the mapping problem.

The basic question is whether load traces that might seem at first glance to be random and unpredictable might have structure that could be exploited by a mapping algorithm. Interestingly, our preliminary results suggest that load traces do indeed have some structure in the form of clearly identifiable properties. Here is a summary of our results:

(1) Except for four machines, all of the systems have low mean loads. However, load can vary drastically. The standard deviation of load (an absolute measure of variation) is typically quite high and grows with increasing mean load. However, the coefficient of variation (a relative measure of variation) actually shrinks with increasing mean load. Further, while desktop machines tend to have lower absolute variation, they have significantly higher relative variation than the other classes. Similarly, maximum load increases with mean load, while maximum load relative to mean load decreases with mean load. Finally, while load maximums don't vary much across machines, desktop machines have much higher maximums relative to their mean loads.

(2) All of the traces exhibit a high degree of self-similarity with Hurst parameters ranging from .63 to .97, strongly biased toward the top of that range. This tells us that load varies in complex ways on all time scales. As a result, lessening the effects of load behavior by smoothing (not migrating long-running tasks, for example) may not be very effective. Another implication is that load may be difficult to model well without many parameters. For our traces, the Hurst parameter measure correlates with the mean and standard deviation of load, but is anti-correlated with maximum load relative to mean load.

(3) The traces also display what we call epochal behavior: the local frequency content of the load signal remains quite stable for long periods of time (150-450 seconds mean) and changes abruptly at epoch boundaries. This suggests that the problem of forecasting load may be able to be decomposed into a sequence of smaller subproblems.

	Mean Load	Sdev Load	COV Load	Max Load	Max/Mean Load	Mean Epoch	Sdev Epoch	COV Epoch	Hurst Param
Mean Load	1.00								
Sdev Load	0.53	1.00							
COV Load	-0.49	-0.22	1.00						
Max Load	0.60	0.18	-0.32	1.00					
Max/Mean Load	-0.36	-0.39	0.51	0.03	1.00				
Mean Epoch	-0.04	-0.10	-0.19	0.08	-0.05	1.00			
Sdev Epoch	-0.02	-0.10	-0.20	0.09	-0.06	0.99	1.00		
COV Epoch	0.07	-0.11	-0.23	0.15	-0.02	0.95	0.96	1.00	
Hurst Param	0.45	0.58	-0.21	0.03	-0.49	0.08	0.10	0.18	1.00

Figure 1: Correlation coefficients (CCs) between all of the discussed statistical properties. The highlighted cells are particularly interesting.

2 Measurement methodology

The load on a Unix system at a given instant is the number of processes that are running or are ready to run. This number is sampled at some rate and the samples over some interval are exponentially averaged to produce a load average which can be accessed from a user program.

We developed a small tool to sample the Digital Unix (DUX) one minute load average at one second intervals and log the resulting time series to a data file. The 1 Hz sample rate was arrived at by subjecting DUX systems to varying loads and sampling at progressively higher rates to determine the rate at which DUX actually updated the value. DUX updates the value at a rate of $1/2$ Hz, thus we chose a 1 Hz sample rate. We ran this trace collection tool on 38 hosts belonging to the Computing, Media, and Communication Laboratory (CMCL) at CMU and the Pittsburgh Supercomputing Center (PSC) for slightly more than one week in late August, 1997. All of the hosts were DEC Alpha DUX machines, and they form four classes:

- *Production Cluster*: 13 hosts of the PSC's "Supercluster", including two front-end machines (axpfea, axpfeb), four interactive machines (axp0 through axp3), and seven batch machines scheduled by a DQS [6] variant (axp4 through axp10.)
- *Research Cluster*: eight machines in an experimental cluster in the CMCL (manchester-1 through manchester-8.)
- *Compute servers*: two high performance large memory machines used by the CMCL group as compute servers for simulations and the like (mojave and sahara.)
- *Desktops*: 15 desktop workstations owned by members of the CMCL (aphrodite through zeno.)

3 Statistics

This section summarizes some basic statistical properties of the load traces. For reference, Figure 1 summarizes the correlation coefficients (CCs) between each of the properties discussed in the paper. Each cell of the table is the CC between the row property and the column property, computed over the 38 load traces. We will refer back to the highlighted cells throughout the paper.

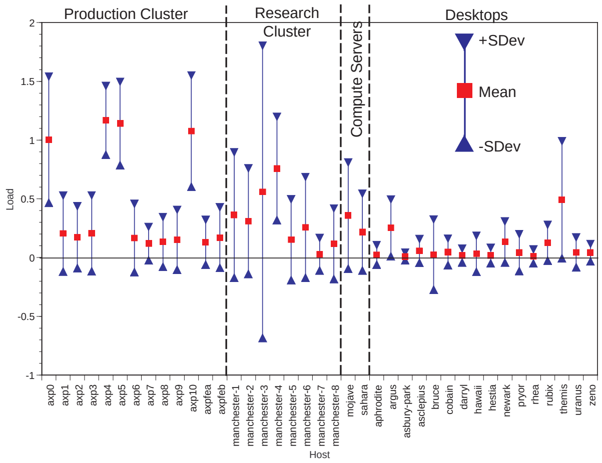

Figure 2: Mean load +/- one standard deviation.

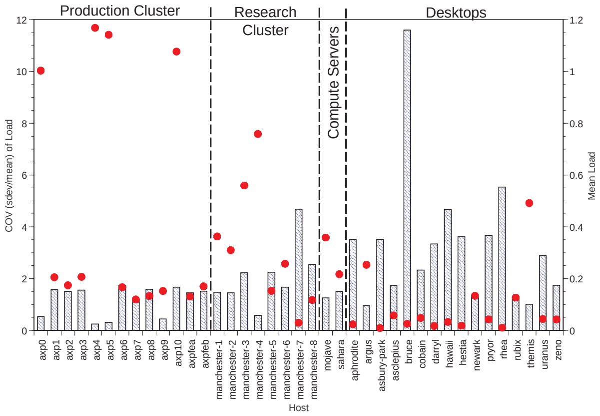

Figure 3: COV of load and mean load.

One would expect that the mean load on clusters and servers is typically higher than that on workstation machines, and our measurements agree. Figure 2 shows the mean load and the +/- one standard deviation points for each of the traces. Clearly, our desktop machines are significantly less loaded than the other machines. Interestingly, we also see a lack of uniformity within each class, despite the long duration of the traces. This is most clear among the Production Cluster machines, where four machines seem to be doing most of the work. This lack of uniformity even over long time scales shows clear opportunity for load balancing or resource management systems.

From Figure 2 we can also see that desktop machines have smaller standard deviations than the other machines.

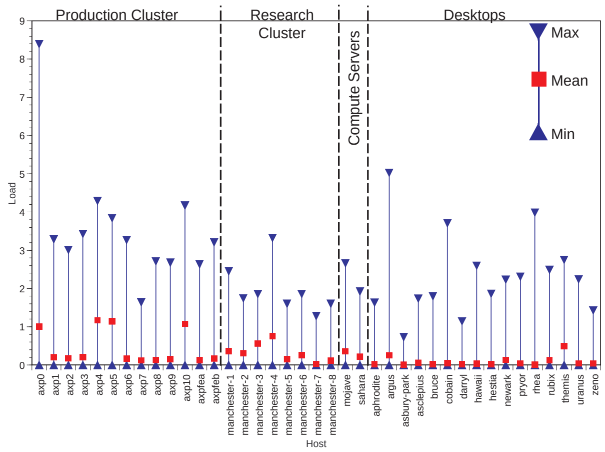

Figure 4: Minimum, maximum, and mean load.

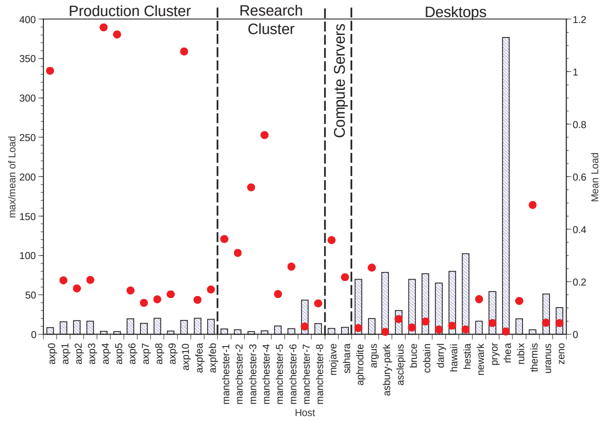

Figure 5: Maximum to mean load ratios and mean load.

What is actually happening is that the standard deviation, which shows how much load varies in *absolute* terms, is reasonably strongly correlated with mean load (CC=0.53 from Figure 1.) However, this is not true in *relative* terms as measured by the coefficient of variation (the standard deviation divided by the mean, abbreviated as the COV.) This can be seen in Figure 3, which plots the COV and the mean load for each of the load traces. Here we can see that desktop machines, with their smaller mean loads, have large COVs compared to the other classes of machines. The CC between mean load and the COV of load is -0.49. As load increases, it varies *less* in relative terms and *more* in absolute terms.

This difference between absolute and relative behavior also holds true for the maximum load. Figure 4 shows the minimum, maximum, and mean load for each of the traces. The minimum load is, not surprisingly, zero in every case. Although it is hard to tell from the graph, maximum load is actually correlated to mean load (CC=0.60 in Figure 1.) Figure 5 plots the ratio max/mean and the mean load for each of the traces. It is clear that this relative measure is inversely related to mean load, and Figure 1 shows that the CC is -0.36. It is also important to notice that while the differences in maximum load between the hosts are rather small (Figure 4), the differences in the max/mean ratio can be quite large (Figure 5.) Desktops are clearly more surprising machines in relative terms.

With respect to the mapping problem, the implication of the differences between relative and absolute measures of variability is that lightly loaded (low mean load) hosts are not always preferable over heavily loaded hosts. For example, if the performance metric is itself a relative one (that the execution time not vary much relative to the mean execution time, say), then a more heavily loaded host may be preferable.

4 Self-similarity

The key observation of this section is that each of the load traces exhibits a high degree of self-similarity. This is significant for two reasons. First, it means that load varies significantly across all time-scales — it is not the case that increasing smoothing of the load quickly tames its variance. A job will have a great deal of variance in its running time regardless of how long it is. Second, it suggests that load is difficult to model and forecast well. In particular, self-similarity is indicative of long memory, possibly non-stationary stochastic processes, and fitting such models to data typically requires a large number of parameters.

Intuitively, a self-similar signal is one that looks similar on different time scales. If a self-similar signal looks jagged over a 10 second time scale, it will also look jagged over a 1 second or 100 second time scale. Another way of visualizing this is to imagine smoothing the signal with an increasingly long window. While the smoothing will quickly blur small variations in the signal, the signal will strongly resist becoming uniform (constant) with increasing window size. Statistically, the variance in increasingly smoothed versions of a series resulting from sampling the signal will only slowly decline. Contrast this with a series derived from picking from a typical stationary probability distribution. The variance of such a series will decline quickly with smoothing.

Self-similarity is more than intuition — it is a well defined mathematical statement about the relationship of the autocorrelation functions of increasingly smoothed versions of certain kinds of long-memory stochastic processes. These stochastic processes model the sort of the mechanisms that give rise to self-similar signals. We shall avoid a mathematical treatment here, but interested readers may want to consult [7] or [8] for a treatment in the context of networking or [2] for its connection to fractal geometry.

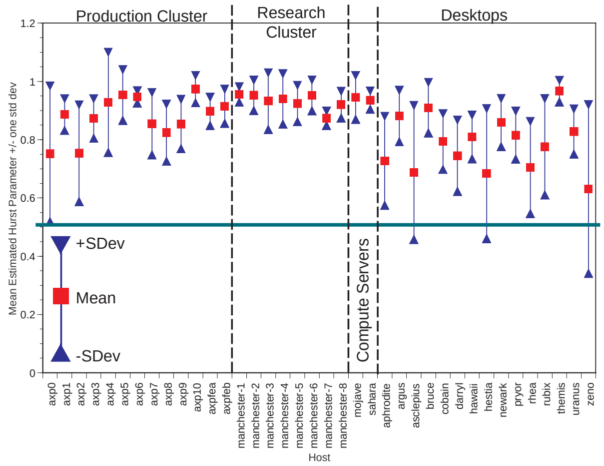

Figure 6: Mean Hurst parameter estimates for traces +/- one standard deviation.

The degree and nature of the self-similarity of a sequence is summarized by the Hurst parameter, H, which ranges from zero to one [4]. $H = 0.5$ indicates no self-similarity and is expected of a series generated from a stationary probability distribution where a sample is completely independent of all previous samples. $H > 0.5$ indicates self-similarity with positive near neighbor correlation, while $H < 0.5$ indicates self-similarity with negative near neighbor correlation.

We examined each of the 38 load traces for self-similarity and estimated each one's Hurst parameter. Unfortunately, there is no consensus on how to best estimate the Hurst parameter of a series. The most common technique is to use several Hurst parameter estimators and try to find agreement among them. The four Hurst parameter estimators we used were R/S analysis, the variance-time method, dispersional analysis, and power spectral analysis. A description of these estimators as well as several others may be found in [2]. *The full paper will describe these and MLE-based estimators here.*

Figure 6 presents our estimates of the Hurst parameters of each of the 38 load traces. In the graph, each central point is the mean of the four estimates, while the outlying points are at +/- one standard deviation. Notice that for small H values, the standard deviation is high due to the inaccuracy of dispersional analysis. The mean Hurst estimates are all significantly above the $H = 0.5$ line and except for three traces with low H values, their +/- one standard deviation points are also above the line.

The traces exhibit self-similarity with Hurst parameters ranging from 0.63 to 0.97, with a strong bias toward the top of that range. On examination of the correlation coefficients (CCs) of Figure 1, we can see that the Hurst parameter has some correlation with mean load (CC=0.45), standard deviation of load (CC=0.58), and is inversely correlated with the max/mean load ratio (CC=-0.49). The latter seems somewhat surprising.

As we discussed above, self-similarity has implications for load modeling and for load smoothing. The long memory stochastic process models that can capture self-similarity tend to have many parameters and are difficult to fit to data. Smoothing the load (mapping large units of computations instead of small units) may be misguided since variance may not decline with increasing smoothing intervals as quickly as quickly as expected.

5 Epochal behavior

The key observation in this section is that while load changes in complex ways, the manner in which it changes remains relatively constant for relatively long periods of time. We refer to a period of time in which this stability holds true as an epoch. For example, the load signal could be a 0.25 Hz Sin wave for a minute and a 0.125 Hz sawtooth wave the next minute — each minute is an epoch. That these epochs exist and are long is significant because it suggests that modeling load can be simplified by modeling epochs separately from modeling the behavior within an epoch. Similarly, it suggests a two stage forecasting process.

The spectrogram representation of a load trace immediately highlights the epochal behavior we discuss in this section. A spectrogram combines the frequency domain and time domain representations of a time series. It shows how the frequency domain changes locally (for a small segment of the signal) over time. For our purposes, this local frequency domain information is the "manner in which [the load] changes" to which we referred earlier. To form a spectrogram, we slide a window of length w over the series, and at each offset k, we Fourier-transform the w elements in the window to give us w complex Fourier coefficients. Since our load series is real-valued, only the first $w/2$ of these coefficients are needed. We form a plot where the x coordinate is the offset k, the y coordinate is the coefficient number, $1, 2, \ldots, w/2$ and the z coordinate is the magnitude of the coefficient. To simplify presentation, we collapse to two dimensions by mapping the logarithm of the z coordinate (the magnitude of the coefficient) to color. The spectrogram is basically a midpoint in the tradeoff between purely time-domain or frequency-domain representations. Along the x axis we see the effects of time and along the y axis we see the effects of frequency.

Figure 7 shows a representative case, a 24 hour trace from the PSC host axp7, taken on August 19, 1997. The top graph shows the time domain representation, while the bottom graph shows the corresponding spectrogram rep-

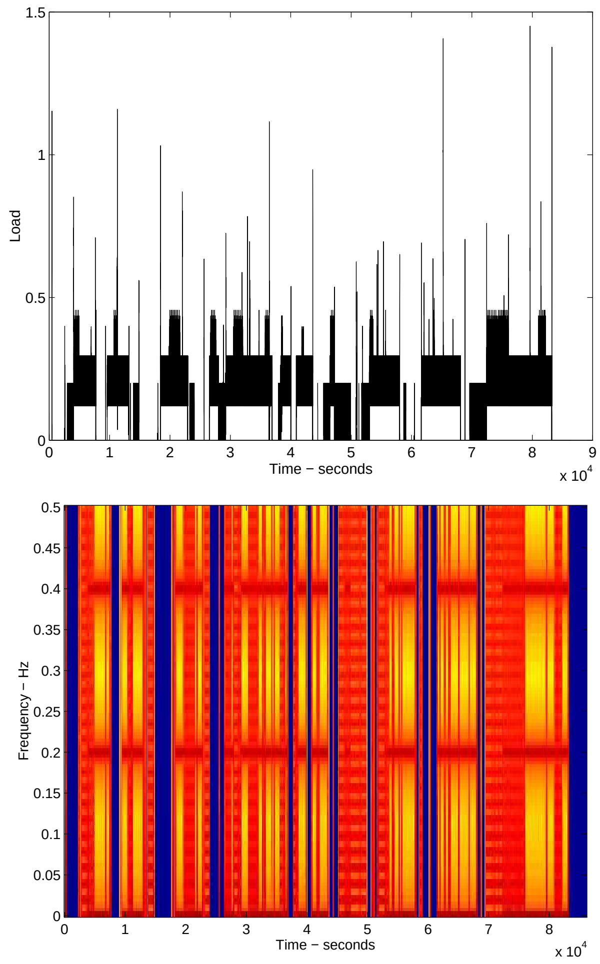

Figure 7: Time domain and spectrogram representations of load for host axp7 on August 19, 1997.

resentation. [1] What is important to note (and which occurs in all the spectrograms of all the traces) are the relatively wide vertical bands. These indicate that the frequency domain of the underlying signal stays relatively stable for long periods of time. We refer to the width of a band as the duration of that frequency epoch.

That these epochs exist can be explained by programs executing different phases, programs being started and shut down, and the like. The frequency content within an epoch contains energy at higher frequencies because of events that happen on smaller time-frames, such as user input, device driver execution, and daemon execution.

We can algorithmically find the edges of these epochs by computing the difference in adjacent spectra in the

[1] Notice that there is evidence of foldover from a too-low sampling rate. We are convinced that the problem lies in how often DUX actually updates its load value and not in our sampling. One explanation is that DUX does not update the load average whenever the length of the ready queue changes. Another is that it does not update it when running device drivers or other in-kernel tasks.

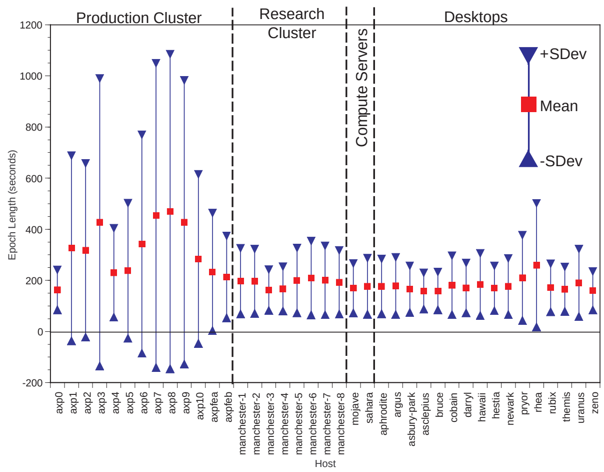

Figure 8: Mean epoch length +/- one standard deviation.

spectrogram and then looking for those offsets where the differences exceed a threshold. Specifically, we compute the sum of mean squared errors for each pair of adjacent spectra. The elements of this vector are compared to an epsilon (5% here) times the mean of the vector. Where this threshold is exceeded, a new epoch is considered to begin. Having found the epochs, we can examine their statistics. Figure 8 shows the mean epoch length and the +/- one standard deviation levels for each of the load traces. The mean epoch length ranges from about 150 seconds to over 450 seconds, depending on which trace. The standard deviations are also relatively high (80 seconds to over 600 seconds.) It is the Production Cluster class which is clearly different when it comes to epoch length. The machines in this class tend to have much higher means and standard deviations than the other machines. One explanation might be that most of the machines run batch-scheduled scientific jobs which may well have longer computation phases and running times. However, two of the interactive machines also exhibit high means and standard deviations. Interestingly, there is no correlation of the mean epoch length and standard deviation to the mean load or to the Hurst parameter, as can be seen in Figure 1.

The standard deviations of epoch length in Figure 8 give us an absolute measure of the variance of epoch length. Figure 9 shows the coefficient of variance (COV) of epoch length and mean epoch length for each trace. The COV is our relative measure of epoch length variance. Unlike with load (Section 3), these absolute and relative measures of epoch length variance are *both* positively correlated with the mean epoch length. In addition, the correlation is especially strong (CC=0.99 for standard deviation and CC=0.95 for COV). As epoch length increases, it varies more in both absolute and relative terms. The statistical properties of epoch lengths are independent of the

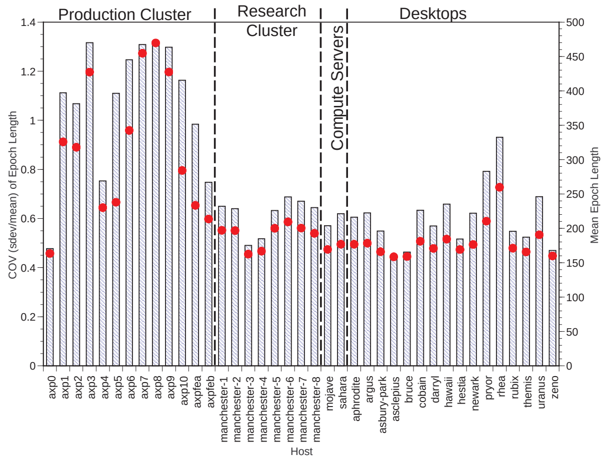

Figure 9: COV of epoch length and mean epoch length.

statistical properties of load.

The implication of long epoch lengths is that the problem of forecasting load may be able to decomposed into a sequence of smaller subproblems, where each subproblem forecasts load within an epoch.

Lack of seasonality

Interestingly, our traces do not show any evidence of seasonality [3]. *The full paper will elaborate.*

6 Conclusions

We have have generated load traces based on measurements of host loads in a real distributed. Analysis of the load traces revealed the following properties of load:

(1) Relative and absolute measures of the variability of load are related differently with mean load. This implies that lightly loaded (low mean load) hosts are not always preferable over heavily loaded hosts. For example, if the performance metric is itself a relative one (that the execution time not vary much relative to the mean execution time, say), then a more heavily loaded host may be preferable.

(2) Load is self-similar, which means that it behaves in complex ways on all timescales. The long-memory stochastic process models that can capture this property are difficult to fit to data and tend to have many parameters. Self-similarity also implies that load smoothing by mapping large units of computation may be misguided. The variance in load may not decline as quickly as expected with increasing scheduling granularity.

(3) The local frequency content of the load signal remains quite stable for long periods of time (epochs) and changes abruptly at epoch boundaries. This suggests that the problem of forecasting or modeling load may be partitionable into a sectioning problem (finding the epochs) and a, hopefully simpler, time series forecasting problem (modeling the behavior of load within the epoch.)

References

[1] ARABE, J., BEGUELIN, A., LOWEKAMP, B., E. SELIGMAN, M. S., AND STEPHAN, P. Dome: Parallel programming in a heterogeneous multi-user environment. Tech. Rep. CMU-CS-95-137, Carnegie Mellon University, School of Computer Science, April 1995.

[2] BASSINGTHWAIGHTE, J. B., BEARD, D. A., PERCIVAL, D. B., AND RAYMOND, G. M. Fractal structures and processes. In *Chaos and the Changing Nature of Science and Medicine: An Introduction* (April 1995), D. E. Herbert, Ed., no. 376 in AIP Conference Proceedings, American Institute of Physics, pp. 54–79.

[3] BROCKWELL, P. J., AND DAVIS, R. A. *Introduction to Time Series and Forecasting*. Springer-Verlag, 1996.

[4] HURST, H. E. Long-term storage capacity of reservoirs. *Transactions of the American Society of Civil Engineers 116* (1951), 770–808.

[5] JENSEN, E. D. A real-time manifesto. http://www.realtime-os.com, 1996.

[6] KAPLAN, J. A., AND NELSON, M. L. A comparison of queueing, cluster, and distributed computing systems. Tech. Rep. NASA TM 109025 (Revision 1), NASA Langley Research Center, June 1994.

[7] LELAND, W. E., TAQQU, M. S., WILLINGER, W., AND WILSON, D. V. On the self-similar nature of ethernet traffic. In *Proceedings of ACM SIGCOMM '93* (September 1993).

[8] MORIN, P. R. The impact of self-similarity on network performance analysis. Tech. Rep. Computer Science 95.495, Carleton University, December 1995.

[9] OBJECT MANAGEMENT GROUP. Realtime corba: A white paper. http://www.omg.org, December 1996. In Progess.

[10] POLZE, A., WERNER, M., AND FOHLER, G. Predictable network computing. Tech. rep., Humboldt University of Berlin, 1996.

[11] RINARD, M., SCALES, D., AND LAM, M. Jade: A high-level machine-independent language for parallel programming. *IEEE Computer 26*, 6 (June 1993), 28–38.

[12] SIEGELL, B., AND STEENKISTE, P. Automatic generation of parallel programs with dynamic load balancing. In *Proceedings of the Third International Symposium on High-Performance Distributed Computing* (August 1994).