On Resource Volatility in Enterprise Desktop Grids

Derrick Kondo[1] Gilles Fedak[1] Franck Cappello[1] Andrew A. Chien[2] Henri Casanova[3]

[1]Laboratoire de Recherche en Informatique/INRIA Futurs
[2]Intel Research
[3]Dept. of Information and Computer Sciences, University of Hawai'i at Manoa

Abstract

Desktop grids, which use the idle cycles of many desktop PC's, are currently one of the largest distributed systems in the world. Despite the popularity and success of many desktop grid projects, the volatility of hosts within desktop grids has been poorly understood. Yet, such host characterization is essential for accurate simulation and modelling of such platforms. In this paper, we present application-level traces of four enterprise desktop grids with a wide range of user bases. We then describe aggregate and per host statistics that reflect the volatility of desktop grid resources. Further, we determine the correlation of volatility between resources, and investigate the correlation of volatility and other host characteristics. Finally, we detail a number of implications of these findings with respect to application performance.

1 Introduction

Desktop grids use the idle cycles of mostly desktop PC's to support large-scale computation and data storage. Despite the popularity and success of many desktop grid projects, the volatility of the hosts within desktop grids has been poorly understood. Yet, this characterization is essential for accurate simulation and modelling of such platforms. In an effort to characterize these types of systems, we gathered availability traces from four real enterprise desktop grids, and using this data, we characterized the volatility of each system.

Specifically, the contributions of this paper are as follows. First, we have made our traces publicly available at `http://desktopgrid.lri.fr/traces`. These data are useful for a number of purposes, including trace-driven simulation and the construction of models. Second, we determine aggregate and per-host statistics of hosts that reflect their volatility. These statistics can be applied to performance modelling and the development of effective scheduling heuristics, for example. Third, we investigate the correlation of volatility between hosts. Finally, we determine the correlation of volatility with other host characteristics, namely host clock rates. Throughout our discussion, we detail a number of implications of these characterization findings with respect to desktop grid applications.

This material is based upon work supported by the National Science Foundation under Grant ACI-0305390.

2 Background

At a high level, a typical desktop grid system consists of a *server* from which tasks are distributed to a *worker* daemon running on each participating host. The worker daemon runs in the background to control communication with the server and task execution on the host, while monitoring the activity on the machine. The worker has a particular recruitment policy used to determine when a task can execute, and when the task must be suspended or terminated. The recruitment policy consists of a *CPU threshold*, a *suspension time*, and a *waiting time*. The CPU threshold is some percentage of total CPU use for determining when a machine is considered idle. Factors that can affect the machine's idle state include mouse or keyboard activity, or other user processes. The suspension time refers to the duration that a task is suspended when the host becomes non-idle. If a host is still non-idle after the suspension time expires, the task is terminated, causing its execution to "fail". When a busy host becomes available again, the worker waits for a fixed period of time of quiescence before starting a task; this period of time is called the waiting time.

Accurate resource characterization can be achieved by obtaining detailed availability traces of the underlying resources with respect to the workers' recruitment policy. The term "availability" has different meanings in different contexts and must be clearly defined for the problem at hand [2]. In the characterization of these data sets, we distinguished among three types of availability:

1. *Host availability.* This is a binary value that indicates whether a host is reachable, which corresponds to the definition of availability in [2]. Causes of host unavailability include power failure, or a machine shutoff, reboot, or crash.

2. *CPU availability.* This is a percentage value that quantifies the fraction of the CPU that can be exploited by a desktop grid application, which corresponds to the definition in [6, 14]. Factors that affect CPU availability include system and user-level compute-intensive processes.

3. *Task execution availability.* This is a percentage value that indicates whether a task can execute on the host or not, according to a desktop grid worker's recruitment policy, and if the task can execute, the percentage of the CPU the task receives. We refer to task execution availability as *exec availability* in short. Causes of exec unavailability include prolonged user keyboard/mouse activity, or a user compute-bound process.

Clearly, exec availability implies CPU availability, which implies host availability. However, the converse is not true. That is, host availability does not imply CPU availability, nor does CPU availability imply exec availability. For example, a host's CPU could be 50% available, but the desktop grid worker would disallow the execution of a desktop grid task to prevent obtrusiveness, and thus the host would be unavailable in terms of exec availability.

3 Related Work on Resource Measurements and Modelling

Although a plethora of work has been done on the measurement and characterization of host and CPU availability, there are two main deficiencies of this related research. First, the traces do not capture all causes of task failures (e.g., users' mouse/keyboard activity), and inferring task failures and the temporal characteristics of availability from such traces is difficult. Second, the traces be affected by idiosyncrasies of the OS [14, 6] instead of showing the true CPU contention for a running task.

In particular, several traces have been obtained that log host availability throughout time. In [4, 12, 2], the authors use a variety of techniques to essentially ping a set of machines periodically to determine host availability. Using traces that record only host availability for the purpose of modeling desktop grids is problematic because it is hard to relate uptimes to CPU cycles usable by a desktop grid application. Several factors can affect an application's running time on a desktop grid, which include not only host availability but also CPU load and user activity. Thus, traces that indicate only uptime are of dubious use for performance modeling of desktop grids or for driving simulations.

Also, there are numerous data sets containing traces of host load or CPU utilization on groups of workstations. Host load is usually measured by taking a moving average of the number of processes in the ready queue maintained by the operating system's scheduler [6], whereas CPU utilization is often measured by the CPU time or clock cycles per time interval received by each process [13, 14, 3]. Since host load is correlated with CPU utilization, we address both types of studies here.

None of the trace studies referenced above record the various events that would cause application task failures (e.g., keyboard/mouse activity) nor are the data sets immune to OS idiosyncrasies, such as process scheduling. In [9], we conducted a study using a similar method for gathering the traces, which will be discussed in Sections 4. However, the study was limited to a single data set obtained from desktops at the San Diego Supercomputer Center (SDSC). By contrast, this study reports and analyzes three new data sets, which have remarkably different user bases and characteristics compared to the SDSC data set. Moreover, we determine and compare several new statistics (and detail the implications for application performance) on a *per host* basis across all *four* data sets, whereas the previous set was limited to *aggregate statistics* for a *single* data set. (Characterization at the per host level is valuable for scheduling for example because it allows one to differentiate one host from another for resource selection or exclusion purposes.)

4 Trace Method

In contrast to the previous studies, we gather traces by submitting measurement tasks to a desktop grid system that are perceived and executed as real tasks. These tasks perform computation and periodically write their computation rates to file. This method requires that no other desktop grid application be running, and allows us to measure exactly the compute power that a real, compute-bound application would be able to exploit. Our measurement technique differs from previously used methods in that the measurement tasks consume the CPU cycles as a real application would.

During each measurement period, we keep the desktop grid system fully loaded with requests for our CPU-bound, fixed-time length tasks, most of which were around 10 minutes in length. The desktop grid worker running on each host ensured that these tasks did not interfere with the desktop user and that the tasks were suspended/terminated as necessary; the resource owners were unaware of our measurement activities. Each task of fixed time length consists of an infinite loop that performs a mix of integer and floating point operations. A dedicated 1.5GHz Pentium processor can perform 110.7 million such operations per second. Every 10 seconds, a task evaluates how much work it has been able to achieve in the last 10 seconds, and writes this measurement to a file. These files are retrieved by the desktop grid system and are then assembled to construct a time series of CPU availability in terms of the number of operations that were available to the desktop grid application within every 10 second interval.

The main advantage of obtaining traces in this fashion is that the application experiences host and CPU availability exactly as any real desktop grid application would. This method is not susceptible to OS idiosyncrasies because the logging is by done by a CPU-bound task actually running on the host itself. Also, this approach captures all the various causes of task failures, including but not limited to mouse/keyboard activity, operating system failures, and hardware failures, and the resulting trace reflects the temporal structure of availability intervals caused by these failures. Moreover, our method takes into account overhead, limitations, and policies of accessing the resources via the desktop grid infrastructure.

5 Trace Data Sets

Using the previously described method, we collected data sets from three desktop grids. One of these desktop grids consisted of desktop PC's at the San Diego Super Computer Center (SDSC) and ran the commercial Entropia [5] desktop grid software. We refer to the data collected from the SDSC environment as the *SDSC trace* *. The other two

*This data set was first reported previously in [10], but we include the data set here for completeness and comparison among the other data sets.

desktop grids consisted of desktop PC's at the University of Paris South, and ran the open source XtremWeb [7] desktop grid software. The Xtremweb desktop grid incorporated machines from two different environments. The first environment was a cluster used by a computer science research group for running parallel applications and benchmarks, and we refer to the data set collected from this cluster as the *LRI trace*. The second environment consisted of desktop PC's in classrooms used by first-year undergraduates, and we refer to the data set as the *DEUG trace*. Finally, we obtained the traces first described in [1] which were measured using a different trace method and refer to this data set as the *UCB trace* [†]. (We describe this method in Section 4.)

The traces that we obtained from our measurements contain gaps. This is expected as desktop resources become unavailable for a variety of reasons, such as the rebooting or powering off of hosts, local processes using 100% of the CPU, the desktop grid worker detecting mouse or keyboard activity, or user actively pausing the worker. However, we observe that a very large fraction ($\geq$ 95%) of these gaps are clustered in the 2 minute range. The average small gap length is 35.9 seconds on the Entropia grid, and 19.5 seconds on the Xtremweb grid.

After careful examination of our traces, we found that these short gaps occur exclusively in between the termination of a task and the beginning of a new task on the same host. We thus conclude that these small gaps do not correspond to actual exec unavailability, but rather are due to the delay of the desktop grid system for starting a new task. In the SDSC grid, the majority of the gaps are spread in the approximate range of 5 to 60 seconds. In the XtremWeb grid, the majority of the gaps are between 0 to 5 seconds and 40 to 45 seconds in length. The sources of this overhead include various system costs of receiving, scheduling, and sending a task as well as an actual built-in limitation that prevents the system from sending tasks to resources too quickly.

Therefore, these small availability gaps observed in both the Entropia and XtremWeb grids would not be experienced by the tasks of a real application, but only in between tasks. Consequently, we eliminated all gaps that were under 2 minutes in our traces by performing linear interpolation. A small portion of the gaps larger than 2 minutes may be also attributed to the server delay and this means that our post-processed traces may be slightly optimistic.

5.1 SDSC Trace

The first data set was collected using the Entropia DCGrid™desktop grid software system deployed at SDSC for a cumulative period of about 1 month across 275 hosts. For our characterization and simulation experiments, we use the longest continuous period of trace measurements, which was the two-week period between 9/3/03 - 9/17/03. Of the 275 hosts, 30 are used by secretaries, 20 are public hosts that are available in SDSC's conference rooms, 12 are used by system administrators, and the remaining are used by SDSC staff scientists and researchers. All hosts are desktop resources that run different flavors of Windows™. During our experiments, about 200 of the 275 hosts were effectively running the Entropia worker (on the other hosts, the users presumably disabled the worker) and we obtained measurements for these hosts. Their clock rates ranged from 179MHz up to 3.0GHz, with an average of 1.19GHz.

5.2 DEUG and LRI Traces

Another two data sets were collected using the XtremWeb desktop grid software continuously over about a 1 month period (1/5/05 - 1/30/05) on a total of about 100 hosts at the University of Paris-Sud. In particular, Xtremweb was deployed on a cluster (denoted by LRI) with a total of 40 hosts, and a classroom (denoted by DEUG) with 40 hosts respectively. The LRI cluster is used by the XtremWeb research group and other researchers for performance evaluation and running scientific applications. The DEUG classroom hosts are used by first year students. Typically, the classroom hosts are turned off when not in use. Classroom hosts are turned on during weekends only if there is a class on weekends.

The XtremWeb worker was modified to keep the output of a running task even if it had failed, and to return the partial output to the server. This removes any bias due to the failure of a fixed-sized task, and the period of availability logged by the task would be identical to that observed by a real desktop grid application.

Compared to the clock distribution of hosts in the SDSC platform, the hosts in the DEUG and LRI platforms have relatively homogeneous clock rates. A large mode in the clock rate distribution for the DEUG platform occurs at 2.4GHz, which is also the median; almost 70% of the hosts have clock rates of 2.4GHz. The clock rates in the DEUG platform range from 1.6GHz to 2.8GHz. In the LRI platform, a mode in the clock rate distribution occurs at about 2GHz, which is also the median; about 65% of the hosts in have clock rates at that speed. The range of clock rates is 1.5GHz to 2GHz.

5.3 UCB Trace

We also obtained an older data set first reported in [1], which used a different measurement method. The traces were collected using a daemon that logged CPU and keyboard/mouse activity every 2 seconds over a 46-day periodon 85 hosts. The hosts were used by graduate students in the EE/CS department at UC Berkeley. We use the largest continuously measured period between 2/28/94 and 3/13/94. The traces were post-processed to reflect the availability of the hosts for a desktop grid application using the following desktop grid settings. A host was considered available for task execution if the CPU average over the past minute was less than 5%, and there had been no keyboard/mouse activity during that time. A recruitment period of 1 minute was used,

Several new statistics of the SDSC trace presented here have not been reported previously.

[†]While the data set was first reported in [1], we present several new statistics of the data.

i.e., a busy host was considered available 1 minute after the activity subsided. Task suspension was disabled; if a task had been running, it would immediately fail with the first indication of user activity. The reason the UCB data set is usable for desktop grid characterization is because the measurement method took into account the primary factors affecting CPU availability, namely both keyboard/mouse activity and CPU availability, and because a strict recruitment policy was applied.

The clock rates of hosts in the UCB platform were all identical, but of extremely slow speeds. In order to make the traces usable in our simulations experiments, we transform clock rates of the hosts to a clock rate of 1.5GHz, which is a modest and reasonable value relative to the clock rates found in the other platforms, and close to the clock rate of host in the LRI platform.

6 Characterization of Volatility

In this section, we characterize in detail resource volatility. In particular, we will focus on fluctuations in exec availability and in our discussion, the term *availability* will denote exec availability unless noted otherwise. We report and discuss aggregate statistics over all hosts in each platform, and also describe per host statistics.

For the SDSC, DEUG, and UCB machines, we found that the platforms were most volatile during *business hours*, during which the users were most active, and were virtually completely dedicated during non-business hours. As such our analysis focuses only on business hour time periods. After close examination, we determined the times delimiting business hours for the SDSC, DEUG, and UCB platforms were 9AM-5PM, 6AM-6PM, and 10AM-5PM respectively. Regarding the LRI platform, which did not have any interactive users, we make no distinction between non-business hours and business hours.

6.1 Quantifying System Volatility via Task Failure Rates

Based on our trace method that captures the temporal structure of resource availability, it is possible to measure system volatility with respect to a task-parallel application. In particular, we compute the expected task failure rate, i.e., the probability that a host will become unavailable before a task completes, using our traces. The trace of availability for each host consists of a series of availability intervals. Each availability interval (delineated by two consecutive periods of unavailability) consists of a start time, stop time, and number of operations measured during that interval. To calculate the failure rate, we choose hundreds of thousands of random points during the availability intervals. At each point, using trace-driven simulation, we determine whether a task of a given size (i.e., number of operations) would run to completion given that the host was initially available for task execution. If so, we count this trial as a success; otherwise, we count the trial as a failure.

Figure 1(a) shows the expected task failure rates computed during business hours for each platform. Also, the least squares line for each platform is superimposed with a dotted magenta line, and the correlation coefficient ρ is shown. For readability, the x-axis shows task sizes not in number of operations, but in execution time on a dedicated 1.5GHz host, from 5 minutes up to almost 7 hours.

The failure rates of each platform from lowest to highest are LRI, SDSC, DEUG, and UCB. The differences among the UCB, SDSC, and DEUG plots is due to the strictness of each system's recruitment policy. For example, the UCB platform has the strictest policy in determining whether a host is available (i.e., a host had to be 95% available before it became usable), relative to the other systems (e.g. a host has to be 80% available on the SDSC platform to be usable). The LRI plot has the lowest slope because LRI corresponds to a relatively underutilized cluster without interactive users; thus, tasks are infrequently terminated due to other batch jobs and never terminated due to keyboard or mouse activity.

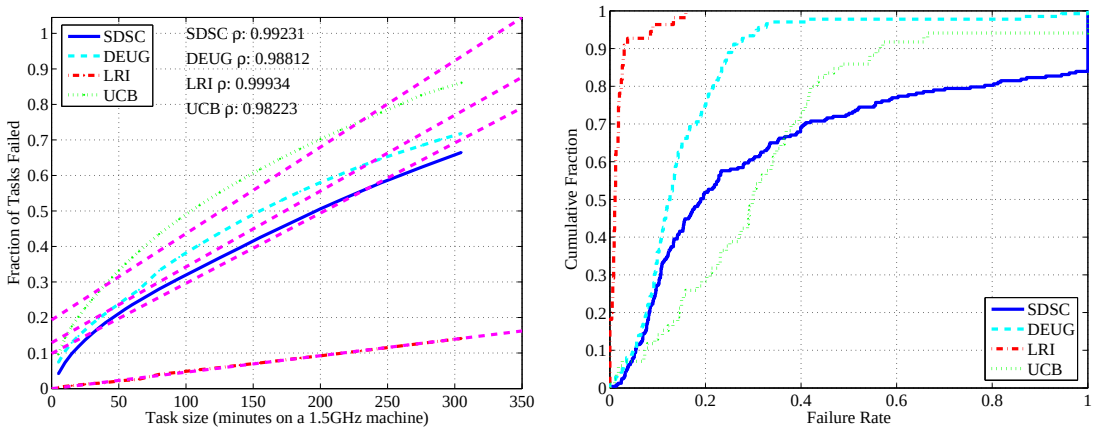
We also find that the expected task failure rate is strongly dependent on the task lengths. (The weekends show similar linear trends, albeit the failure rates are lower.) It appears that in all platforms the task failure rate increases with task size and that the increase is roughly linear; the lowest correlation coefficient is 0.98, indicating that there exists a strong linear relationship between task size and failure rate. (Clearly, as the task size approaches infinity, the task failure rate will eventually plateau as it approaches one. Nevertheless, the relationship is approximately linear for a reasonable range of task sizes.)

One practical way to apply this finding is by incorporating the linear fit in a closed-form model that describes the aggregate performance attainable by a task parallel, high-throughput application on the corresponding desktop grid (see [10] for an example with the SDSC data set). This model in turn can be used by application developers to determine the aggregate system performance that corresponds to a particular task size.

While Figure 1(a) shows the aggregate task failure rate of the system, Figure 1(b) shows the cumulative distribution of the failure rate per host in each platform, using a particular task size of 35 minutes on a dedicated 1.5GHz host. The heavier the tail, the more volatile the hosts in the platform. Overall, the distributions appear quite skewed. That is, a majority of the hosts are relatively stable. For example, with the DEUG platform, about 75% of the hosts have failure rates of 20% or less. The UCB platform is the least skewed, but even so, over 70% of the hosts have failure rates of 40% or less.

Surprisingly, in Figure 1(a), SDSC has lower task failure rates than UCB, yet in Figure 1(b), SDSC has a larger fraction of hosts with failure rates 20% or less compared to UCB. The discrepancy can be explained by the fact that UCB still has a larger fraction of hosts with failure rates 40% or more than SDSC; after averaging, SDSC has lower failure rates.

The fact that a small fraction of hosts have relatively high failure rates can be use in the design of scheduling heuristics for soft real-time applications, for example. One could



(a) Aggregate

(b) Per host for 35 minute tasks

Figure 1. Task failure rates during business hours

develop a set of scheduling mechanisms and heuristics that exclude the most volatile hosts from a computation. While some of the heuristics described in [9] use host exclusion, the heuristics do not use the same measure of volatility for the exclusion criterion; thus, for future work, we will design and evaluate new heuristics based on this finding.

6.2 Correlation of Volatility Between Hosts

In the previous section, we determined the volatility of the system as a whole and per host. Here, we investigate the correlation of volatility between hosts. This is an important issue because it would help determine the feasibility of running certain types of applications (e.g. data-parallel applications, which required inter-node communication) on desktop grids. Moreover, a number of analytical studies in desktop grids assume that exec availability is independently and identically distributed [11, 8] to simplify probability calculations. We investigate the validity of such an assumption.

First, we studied the temporal correlation of exec availability across hosts by calculating the correlation of exec availability between pairs of hosts. Specifically, we compared the availability for each pair of hosts by adding 1 if both machines were available or both machines were unavailable, and subtracting 1 if one host was available and the other one was not. This method was used by the authors in [3] to study the correlation of availability among thousands of hosts at Microsoft Corp.

Figure 2 shows the cumulative fraction of hosts pairs that are below a particular correlation coefficient. The line labelled "trace" in the legend indicates correlation calculated according to the traces. The line labelled "min" indicates the minimum possible correlation given the percent of time each machine is available, and likewise for the line labelled "max".

Overall, we can see that in all the platforms at least 60% of the host pairings had positive correlation, which indicates that a host is often available or unavailable when another host is available or unavailable respectively. However, Figures 2(a) and 2(d) also show that this correlation is due to the fact that most hosts are usually available (for example, 80% of the time, hosts have CPU availability of 80% or higher), which is reflected by how closely the trace line follows the random line in the figure. That is, if two host are available (or unavailable) most of the time, they are more likely to be available (or unavailable) at the same time, even randomly. As a result, the correlation observed in the traces would in fact occur randomly because the hosts have such high availability. This in turn gives strong evidence that the availabilities of hosts in the SDSC and UCB platforms would otherwise be uncorrelated, if it were not for the hosts' high availability.

We believe that the insignificance of correlation of exec availability in the SDSC and DEUG traces is primarily due to the user base of the hosts. As mentioned previously, the user base of the SDSC consisted primarily of research scientists and administrators who we believe used the Windows hosts primarily for interactive tasks (such as word processing and surfing the Internet) that did not directly affect the availability of other hosts. Similarly, for the UCB platform, we believe the students used the hosts primarily for short durations, which is evident by the short unavailability intervals. So, because the primary factors causing host unavailability, i.e., user processes and keyboard/mouse activity, are often independent from one machine to another in desktop environments, we observe that exec availability in desktop grids is often not significantly correlated.

On the other hand, Figures 2(b) and 2(c) show different trends where the line corresponding to the trace differs significantly with respect to correlation (as much as 20%) from the line corresponding to random correlation. The weak correlation of the DEUG trace is due to the particular configuration of the classroom machines. These machines had wake-on-LAN enabled Ethernet adapters, which allowed the machines to be turned on remotely. The system administrators had con-

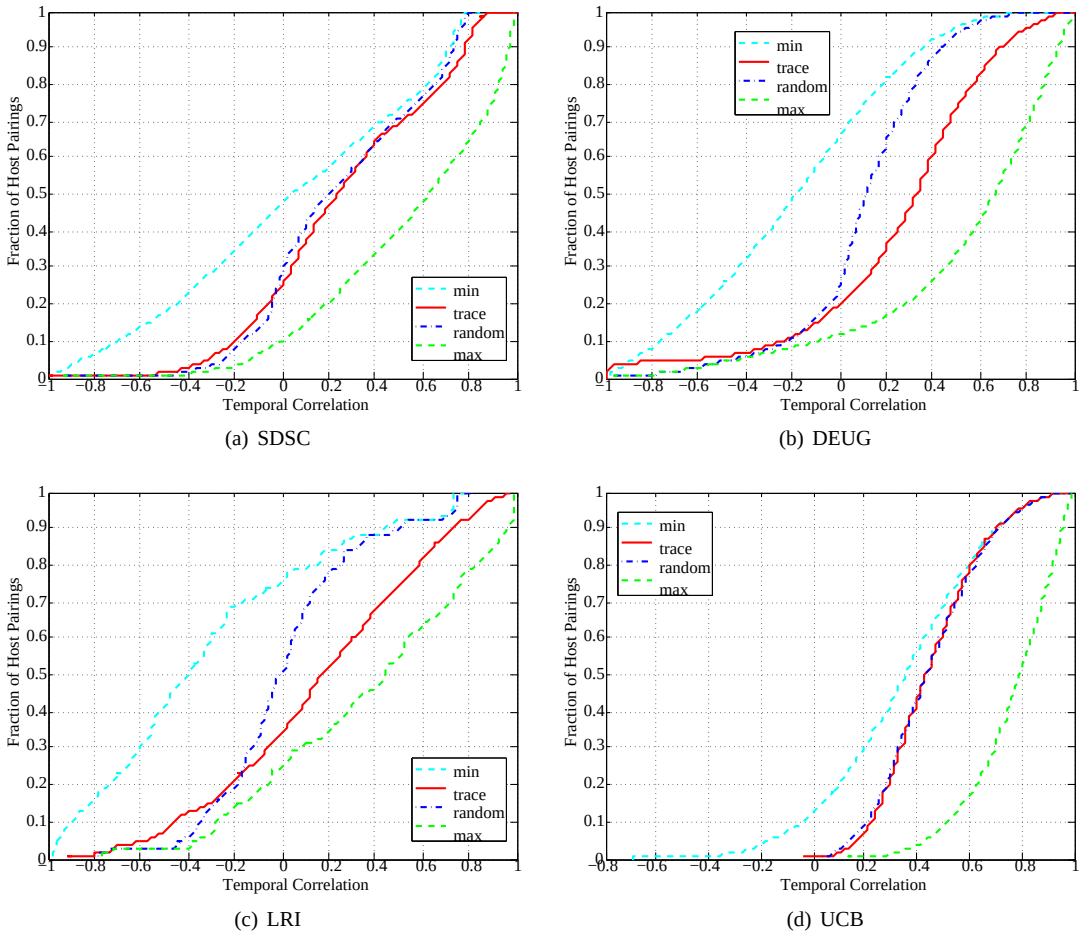


Figure 2. Correlation of availability.

figured these machines to "wake" every hour if they had been turned off by a user. Since most machines are turned off when not in use, many machines were awakened at the same time, resulting in the weak correlation of availability. We believe that this wake-on-LAN configuration is specific to the configuration of the machines in DEUG, and that in general, machine availability is not significantly correlated in desktop grids where keyboard/mouse activity is high.

Hosts in the LRI platform also shows significant correlation relative to random. This behavior is expected as batch jobs submitted to the cluster tend to consume a large number of nodes simultaneously, and consequently, the nodes are unavailable for desktop grid task execution at the same times.

6.3 Correlation of Volatility with Host Clock Rates

In the previous sections, we determined statistics regarding the volatility of platforms, individual hosts in those platforms, and the correlation of volatility between hosts. In addition, we were interested in the relation between volatility and other host characteristics, in particular, host clock rates. This interest was spurred by our desire to find simple, static criteria by which to predict host performance, which would be useful for scheduling, for example.

We hypothesized that a host's clock rate could be a inverse indicator of host volatility. Intuitively, host speed could be correlated with a number of other machine characteristics of volatility. For example, the faster a host's clock rate is, the faster it should complete a task, and the lower the failure rate should be. Or the faster a host's clock rate is, the more often a user will be using that particular host, and the less available the host will be. Surprisingly, host speed is not as correlated with these factors as we first believed.

Table 1 shows the correlation of clock rate with various measures of host volatility for the SDSC trace. (Because the clock rates in the other platforms were roughly uniform, correlation calculations would have been inconclusive.) Since clock rates often increase exponentially over time, we also computed the correlation of the log of clock rates with the other factors and found similar trends. We compute the correlation between clock rate and the mean time per availability interval to capture the relationship between clock rate the temporal structure of availability. However, it could be the case that a host with very small availability intervals is available most of the time. So, we also compute the correlation between clock rate and percent of time each host is unavailable. We find that there is little correlation between the clock

X	Y	ρ
clock rate	mean availability interval length (time)	-0.0174
clock rate	% of time unavailable	0.1106
clock rate	mean availability interval length (ops)	0.5585
clock rate	task failure rate (15 minute task)	-0.2489
clock rate	P(complete 15 min task in 15 min)	0.859

Table 1. Correlation of host clock rate and other machine characteristics during business hours for the SDSC trace

rate and mean length of CPU availability intervals in terms of time (see row 1 in Table 1), or percent of the time the host is unavailable (see row 2). We explain this by the fact that many desktops for most of the time are being used for intermittent and brief tasks, for example for word processing, and so even machines with relatively low clock rates can have high unavailability (for example due to frequent mouse/keyboard activity), which results in availability similar to faster hosts. Moreover, the majority of desktops are distributed in office rooms. So desktop users do not always have the choice of a faster desktop.

What matters more to an application than the time per interval is the operations per interval, how it affects the task failure rate, and whether it is related to the clock rate of the host. We compute the correlation between clock rate and CPU availability in terms of the mean operations per interval and failure rate for a 15 minute task. There is only weak positive correlation between clock rate and the mean number of operations per interval (see row 3), and weak negative correlation between the clock rate and failure rate (see row 4). Any possibility of strong correlation would have been weakened by the randomness of user activity. Nevertheless, the factors are not independent of clock rate because hosts with faster clock rates tend to have more operations per availability interval, thus increasing the chance that a task will complete during that interval.

Furthermore, in row 5, we see the relationship between clock rate and rate of successful task completion within a certain amount of time. In particular, row 5 shows the fairly strong positive correlation between clock rate and the probability that a task completes in 15 minutes or less. The size of the task is 15 minutes when executed on a dedicated 1.5GHz. (We also computed the correlation for other task sizes and found similar correlation coefficients.) Clearly, whether a task completes in a certain amount of time is related to clock rate. However, this relationship is slightly weakened due to the randomness of exec unavailability, as unavailability could cause a task executing on a relatively fast host to fail. One implication of this correlation shown in rows 3-5 is that a scheduling heuristic based on using host clock rates for task assignment may be effective.

The correlation between host clock rate and task failure rate should increase with task size (until all hosts have very high failure rates close to 1). Short tasks that have a very low mean failure rate (near zero) over all hosts will naturally have low correlation. As the task size increases, the failure rate will be more correlated with clock rates since in general the faster hosts will be able to finish the tasks sooner. Table 2 shows the correlation coefficient between clock rate and failure rate for different task sizes. There is only weak negative correlation between host speed and task failure rate, and it increases in general with the task size as expected. Again, we believe the weak correlation is partly due to the randomness of keyboard and mouse activity on each machine. One consequence of this result with respect to scheduling is that the larger the task size, the more important it is to schedule the tasks on hosts with faster clock rates.

Task size	Failure rate	ρ
5	0.063	-0.2053
10	0.097	-0.2482
15	0.132	-0.2489
20	0.155	-0.2624
25	0.177	-0.2739
30	0.197	-0.280
35	0.220	-0.2650

Table 2. Correlation of host clock rate and failure rate during business hours. Task size is in term of minutes on a dedicated 1.5GHz host.

7 Summary and Future Work

We described a simple trace method that measured availability in a way that reflects the same availability that could be experienced by a real application. We used this method to gather data sets on three platforms with distinct clock rate distributions and user types. In addition, we obtained a fourth data set gathered earlier by the authors in [1]. Then, we derived several useful statistics regarding the volatility for each system as a whole and for individual hosts.

The findings of our characterization study can be summarized as follows. First, we found that task failure rates on each system were linearly correlated with task size, within a reasonable range, for data sets with dissimilar user bases ranging from interactive user to users submitting batch jobs; moreover, task failure rates can be approximated by a linear

function of task size, despite the differences among the user scenarios. This allows one to construct a closed-form performance model of each system as a function of task size. Moreover, we determined that only a small fraction of hosts tend to be extremely unstable, and this fact can be used to design effective scheduling heuristics that exclude this fraction of volatile hosts.

Second, we observed that on platforms with interactive users, volatility as measured by exec availability tends to be insignificantly correlated across hosts; at the same time, a signification fraction of the hosts are available simultaneously. However, the significance of correlation can be affected by the configuration of the hosts; for example wake-on-LAN enabled Ethernet adapters can cause correlated availability among hosts. Also, in platforms used to run batch jobs, availability is significantly correlated. These correlation findings are relevant to data parallel applications that require correlated availability to permit inter-node communication and efficient execution.

Third, volatility in terms of availability interval lengths measured in seconds are not correlated with clock rates nor is the percentage of time a host is unavailable. This means that hosts with faster clock rates are not necessarily used more often. Nevertheless, interval lengths in terms of operations and task failure rates are correlated with clock rates. This indicates that the selecting resources according to clock rates may be beneficial. This conclusion is directly relevant to desktop grid scheduling, as it suggest that clock rates themselves are a good indicator of host performance.

Currently, we are obtaining traces on Internet desktop grid environments via the XtremLab project (see `http://xtremlab.lri.fr`). We then hope to extend our characterization to Internet desktop grids environments, and compare and contrast the results to enterprise environments.

Acknowledgments

We gratefully acknowledge Kyung Ryu and Amin Vahdat for providing the UCB trace data set and documentation.

References

[1] R.H. Arpaci, A.C. Dusseau, A.M. Vahdat, L.T. Liu, T.E. Anderson, and D.A." Patterson. The Interaction of Parallel and Sequential Workloads on a Network of Workstations. In *Proceedings of SIGMETRICS'95*, pages 267–278, May 1995.

[2] R. Bhagwan, S. Savage, and G. Voelker. Understanding Availability. In *In Proceedings of IPTPS'03*, 2003.

[3] W. Bolosky, J. Douceur, D. Ely, and M. Theimer. Feasibility of a Serverless Distributed file System Deployed on an Existing Set of Desktop PCs. In *Proceedings of SIGMETRICS*, 2000.

[4] J. Brevik, D. Nurmi, and R. Wolski. Quantifying Machine Availability in Networked and Desktop Grid Systems. Technical Report CS2003-37, Dept. of Computer Science and Engineering, University of California at Santa Barbara, November 2003.

[5] 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*, 63:597–610, 2003.

[6] P. Dinda. The Statistical Properties of Host Load. *Scientific Programming*, 7(3–4), 1999.

[7] G. Fedak, C. Germain, V. N'eri, and F. Cappello. XtremWeb: A Generic Global Computing System. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGRID'01)*, May 2001.

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

[9] D. Kondo, A. Chien, and Casanova H. Rapid Application Turnaround on Enterprise Desktop Grids. In *ACM Conference on High Performance Computing and Networking, SC2004*, November 2004.

[10] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien. Characterizing and Evaluating Desktop Grids: An Empirical Study. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'04)*, April 2004.

[11] Y. Li and M. Mascagni. Improving performance via computational replication on a large-scale computational grid. In *Proc. of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*, May 2003.

[12] D. Long, A. Muir, and R. Golding. A Longitudinal Survey of Internet Host Reliability. In *14th Symposium on Reliable Distributed Systems*, pages 2–9, 1995.

[13] M. Mutka and M. Livny. The available capacity of a privately owned workstation environment . *Performance Evaluation*, 4(12), July 1991.

[14] R. Wolski, N. Spring, and J. Hayes. Predicting the CPU Availability of Time-shared Unix Systems. In *Peoceedings of 8th IEEE High Performance Distributed Computing Conference (HPDC8)*, August 1999.