Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home

Bahman Javadi, *Member, IEEE,* Derrick Kondo, *Member, IEEE,*
Jean-Marc Vincent, *Member, IEEE* and, David P. Anderson, *Member, IEEE*

Abstract—In the age of cloud, Grid, P2P, and volunteer distributed computing, large-scale systems with tens of thousands of unreliable hosts are increasingly common. Invariably, these systems are composed of heterogeneous hosts whose individual availability often exhibit different statistical properties (for example stationary versus non-stationary behavior) and fit different models (for example exponential, Weibull, or Pareto probability distributions). In this paper, we describe an effective method for discovering subsets of hosts whose availability have similar statistical properties and can be modelled with similar probability distributions. We apply this method with about 230,000 host availability traces obtained from a real Internet-distributed system, namely SETI@home. We find that about 21% of hosts exhibit availability that is a truly random process, and that these hosts can often be modelled accurately with a few distinct distributions from different families. We show that our models are useful and accurate in the context of a scheduling problem that deals with resource brokering. We believe that these methods and models are critical for the design of stochastic scheduling algorithms across large systems where host availability is uncertain.

Index Terms—statistical availability models, reliability, resource failures, stochastic scheduling

1 Introduction

With rapid advances in networking technology and dramatic decreases in the cost of commodity computing components, large distributed computing platforms with tens or hundreds of thousands of *unreliable* and *heterogeneous* hosts are common. The uncertainty of host availability in P2P, cloud, or Grid systems can be due to the host usage patterns of users, or faulty hardware or software. Clearly, the dynamics of usage, and hardware and software stacks are often heterogeneous, spanning a wide spectrum of patterns and configurations. At same time, within this spectrum, subsets of hosts with homogeneous properties can exist.

So one could also expect that the statistical properties (stationary versus non-stationary behavior for example) and models of host availability (exponential, Weibull, or Pareto for example) to be heterogeneous in a similar way. That is, host subsets with common statistical properties and availability models can exist, but differ greatly in comparison to other subsets.

The goal of our work is to be able to discover host subsets with similar statistical properties and availability models within a large distributed system. In particular, the main contributions are as follows:

- *Methodology.* Our approach is to use tests for randomness to identify hosts whose availability is independent and identically distributed (iid). For these hosts, we use clustering methods with distance metrics that compare probability distributions to identify host subsets with similar availability models. We then apply parameter estimation for each host subset to identify the model parameters.
- *Modelling.* We apply this method on one of the largest Internet-distributed systems in the world, namely SETI@home. We take availability traces from about 230,000 hosts in SETI@home, and identify host subsets with matching statistical properties and availability models. We find that a significant fraction (21%) of hosts exhibit iid availability, and a few distributions from several distinct families (in particular the Gamma and hyper-exponential) can accurately model the availability of host subsets.
- *Scheduling.* We show the utility and accuracy of these models for the design of stochastic scheduling algorithms for large systems where availability is uncertain. We focus on a resource brokering problem where incoming jobs must be routed to a set of schedulers, each of which oversees a set of unreliable hosts. With respect to scheduling, the results show that a generalized round-robin method performs best in simulation experiments compared to other algorithms, confirming and reinforcing theoretical results in the literature. With respect to modelling, the results show that analytical results based on our stochastic models match those obtained from the availability traces alone, verifying the accuracy of our models.

- B. Javadi and D. Kondo are with INRIA, France. E-mail: {javadi, dkondo}@imag.fr
- JM. Vincent is with the University of Joseph Fourier, France. Email: jean-marc.vincent@imag.fr
- D. Anderson is with UC Berkeley, USA. Email: davea@ssl.berkeley.edu

2 Modelling Approach

2.1 Application context

We describe here the context of our application, which is used to guide what we model. Our modelling is conducted in the context of volunteer distributed computing, which uses the free resources of Internet-distributed hosts for large-scale computation and storage. Currently this platform is limited to embarrassingly parallel applications where the main performance metric is throughput. One of the main research goals in this area is to broaden the types of applications that can effectively leverage this platform.

We focus on the problem of scheduling applications needing fast response time (instead of high throughput). This class of applications includes those with batches of compute-intensive tasks that must be returned as soon as possible, and also those whose tasks are structured as a directed acyclic graph (DAG). For these applications, the temporal structure of availability and resource selection according to this structure is critical for their performance.

2.2 Modeling Process

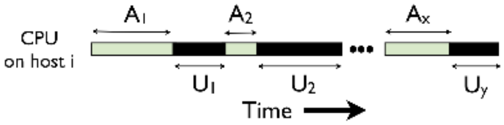


Fig. 1. CPU availability and unavailability intervals on one host

Thus our modelling focuses on the interval lengths of continuous periods of CPU availability or unavailability for individual hosts. We term a continuous interval to be an **(un)availability interval** (see Figure 1). The lengths of the (un)availability intervals of a particular host over time is the process that we model. We model the process of unavailability and the process of availability separately, as each showed significant differences.

The processes of availability interval lengths span a spectrum with completely deterministic behavior on one end and truly random behavior on the other. Some processes are completely deterministic due to periodicity or trends, such as hour-in-day or day-in-week time effects. Others are completely random due to the unpredictability of user or application behavior. Others show a combination of determinism and randomness (for instance, a CPU that is available consistently from 11:00 onward for an uncertain amount of time).

We focus our modelling efforts on hosts whose availability is truly random, as these hosts constitute a significant fraction of the platform (21% in terms of the total number of hosts). Our modelling method is general enough to also include those hosts that exhibit a combination of deterministic and random availability. We can do so by removing what is deterministic and model the random phenomenon that remains. For instance, with a CPU consistently available from 11;00 onward, we can model its availability by focusing on the availability period after 11:00.

The remaining hosts that have purely deterministic availability can be modelled using different methods (see Section 4 for details). There is evidence showing that these hosts are in the minority. For instance, in [24], we showed that only about 6% of all hosts have availability exhibiting repeated diurnal cycles.

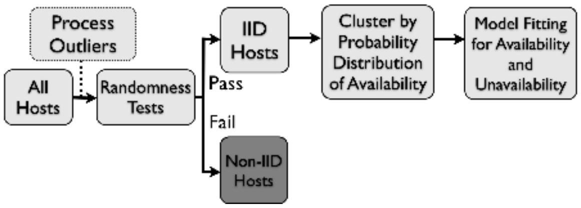

Process Outliers
All Hosts
Randomness Tests
Pass
Fail
IID Hosts
Non-IID Hosts
Cluster by Probability Distribution of Availability
Model Fitting for Availability and Unavailability

Fig. 2. Modelling workflow

To model the random availability of hosts, we use the following approach. Figure 2 summarizes the workflow. We define availability and describe how our availability measurements were gathered in Section 3. We contrast this measurement method and our modelling approach to related work in Section 4. We determine which hosts exhibit random, independent, and stationary availability in Section 6. For hosts that pass the randomness tests, we describe the clustering of hosts by their distribution of availability in Sections 7.0.2- 7.0.3. We describe the results of parameter estimation for each distribution in Section 7.1. Then we evaluate the utility and accuracy of our models for scheduling in Section 8.

3 Measurement Method for CPU Availability

BOINC [2] is a middleware for volunteer distributed computing. It is the underlying software infrastructure for projects such as SETI@home, and runs across over 1 million hosts over the Internet.

We instrumented the BOINC client to collect CPU availability traces from about 230,000 hosts over the Internet between April 1, 2007 to January 1, 2009. We define CPU availability to be a binary value indicating whether the CPU was free or not. The traces record the time when CPU starts to be free and stops.

Our traces were collected using the BOINC server for SETI@home. The traces are not application dependent nor specific to SETI@home because they are recorded at the level of the BOINC client (versus the application it executes). In total, our traces capture about 57,800 years of CPU time and 102,416,434 continuous intervals of CPU availability.

Artifacts due to a measurement method or middleware itself are almost inevitable. In our case, artifacts resulted from a benchmark run periodically every five days by the BOINC client. As a result, availability intervals longer than five days were truncated prematurely and separated by about a one minute period of unavailability before the next availability interval. Histograms and mass-count graphs showed that these 5-day intervals were outliers and significantly skewed the availability interval mass distribution. As such, we preprocessed the traces to remove these gaps after five day intervals artificially introduced by our middleware. This minimized the effect of these anomalies in the modelling process.

4 Related Work

This work differs from related in terms of what is measured and what is modelled. In terms of measurements, this work differs from others by the types of resources measured (home versus only within an enterprise or university), the scale and duration (hundreds of thousands over 1.5 years versus hundreds over weeks), and the type of measurement (CPU availability versus host availability). Specifically, the studies [23], [8] focus on host in the enterprise or university setting and may have different dynamics compared to hosts primarily on residential broadband networks. Our study includes both enterprise and university hosts, in addition to hosts at home.

Also, other studies focus on hundreds of hosts over a limited time span [23], [26], [1]. Thus their measurements are of limited breadth and could be biased to a particular platform. The limited number of samples per host prevents accurate modelling of the individual resource.

Furthermore, while many (P2P) studies of availability exist [28], [7], [30], these studies focus on host availability, i.e., a binary value indicating whether a host is reachable. By contrast, we measure CPU availability as defined in Section 3. This is different as a host can clearly be available but not its CPU. Nevertheless, host availability is subsumed by CPU availability.

In terms of modelling, most related works [27], [35], [34], [13] focus on modelling the system as a whole instead of individual resources, or the modelling does not capture the temporal structure of availability [3], [13], [24], [18], [16]. Yet this is essential for effective resource selection and scheduling [29], [6]. For instance, in [27], the authors find that availability in wide-area distributed systems can be best modelled with a Weibull or hyperexponential distribution. However, these models consider the system as a whole and are not accurate for individual resources.

For another instance, in our own past work in [24], we conducted a general characterization of SETI@home. However, we ignored availability intervals by taking averages to represent each host's availability. So we used a different data representation with different distance metrics when clustering. Moreover, the purpose was different because we intentionally focused on identifying *correlated*, deterministic patterns instead of random ones. As such, we did not build statistical models of availability intervals, and did not address issues such as partitioning hosts according to randomness and probability distributions.

5 Experimental Setup

We conduct all of our statistical analysis below using Matlab 2009a on a 32-bit on a Xeon 1.6GHz server with about 8.3GB of RAM. We use when possible standard tools provided by the Statistical Toolbox. Otherwise, we implement or modify statistical functions ourselves.

6 Randomness Testing

A fraction of hosts are likely to exhibit availability intervals that contain trends, periodicity, or non-stationarity. For these hosts, modelling their availability using a probability distribution is difficult given the change of its statistical properties over time. Therefore, as the preliminary phase before data analysis and modeling, we apply randomness tests to determine which hosts have truly random availability intervals.

We conduct three well-known non-parametric tests, namely the runs test, runs up/down test, and Kendall-tau test [32], [9]. For all tests, the availability intervals of each hosts was imported as a given sequence or time series; the same was done with unavailability intervals.

We describe intuitively what the tests measure. The runs test compares each interval to the mean. The runs up/down test compares each interval to the previous interval to determine trends. The Kendall-tau test compares the length of an interval with all previous interval lengths in the sequence.

Since the hypothesis for all of these tests are based on the normal distribution, we must make sure that there are enough samples for each host, i.e., at least 30 samples, according to [10] and personal communication with a statistician. About 20% of hosts do not have enough samples because of a limited duration of trace measurement. (Measurement for a host began only after the user downloaded and installed the instrumented BOINC client. So some hosts may have only a few measurements because they began using the instrumented BOINC client only moments before we collected and ended the trace measurements). So we ignore these hosts in our statistical analysis. Finally, we apply all three tests on 168,751 hosts with a significance level of 0.05.

As there is no perfect test for randomness, we decide to apply all tests and to consider only those hosts that pass all three tests to be conservative. Table 1 shows the fraction of hosts that pass these tests. Each subcolumn of a particular test corresponds to either a sequence of availability or unavailability. For instance, the availability sequence of 101,649 hosts passed the runs std test, and the unavailability sequence of 121385 hosts passed the same test. In total, 35,686 hosts have availability and unavailability sequences that pass all three tests.

While iid hosts may not be in the majority (i.e., 21%), together they still form a platform with significant computing power. For example, the project FOLDING@home alone provides about 8.127 PetaFLOPS of performance. The hosts with iid (un)availability thus contribute about 1.7 PetaFLOPS, which is significant. Moreover, as discussed in Section 2.2, our modelling techniques could be extended to handle hosts with a mixture of deterministic and random behavior.

7 Clustering by Probability Distribution

For hosts whose (un)availability is truly random, our approach was to first inspect the distribution using histograms and the mass-count disparity (see Figure 3). We observe significant right skew. For example, the shortest 80% of the availability intervals contribute only about 10% of the total fraction of availability. The remaining longest 20% of intervals contribute about 90% of the total fraction of availability. The distribution of unavailability has a much heavier tail compared with availability. The implication for modelling is that we should focus on the larger availability intervals. But which hosts can be modelled by which distributions with which parameters?

7.0.1 Clustering Background

We use two standard clustering methods, in particular k-means [14] and hierarchical clustering, to cluster hosts by their distribution. K-means randomly selects k cluster centers, groups each point to the nearest cluster center, and repeats until convergence. The advantage is that it is extremely fast. The disadvantages are that it requires k to be specified a priori, the clustering results depend in part on the cluster centers chosen initially, and the algorithm may not converge.

Hierarchical clustering iteratively joins together the two sub-clusters that are closest together in terms of the average all-pairs distance. It starts with the individual data points working its way up to a single cluster. The advantage of this method is that one can visualize the distance of each sub-cluster at different levels in the resulting dendrogram, and that it is guaranteed to converge. The disadvantage is that we found the method to be 150 times slower than k-means.

7.0.2 Distance Metrics for Clustering

We tested several distance metrics for clustering, each of which measures the distance between two CDFs [11]. Intuitively, Kolmogorov-Smirnov determines the maximum absolute distance between the two curves. Kuiper computes the maximum distance above and below of two CDFs. Cramer-von Mises finds the difference between the area under the two CDFs. Anderson-Darling is similar to Cramer-von Mises, but has more weight on the tail.

Using these distance metrics is challenging when the number of samples is too low or too high. In the former case, we do not have enough data to have confidence in the result. In the latter case, the metric will be too sensitive. A model by definition is only an approximation. When the number of samples is exceedingly high, the distance reported by those metrics will be exceedingly high as well. This is because the distance is often a multiplicative factor of the number of samples.

Our situation is complicated further because we have a different number of samples per host. Ideally, when computing each distance metric we should have the same number of samples. Otherwise, the distance between two hosts with 1000 samples will be incomparable to the distance between two hosts with 100 samples.

Our solution is to select a fixed number of intervals from each host *at random*. This method was used also in [27], where the authors had the same issue. We also discussed this issues with a statistician, who confirmed that this is an acceptable approach.

However, if we choose the fixed number to be too high, it will exclude too many hosts from our clustering. If we choose the number to be too low, we will not have statistical confidence in the result. We choose the fixed number to be 30 intervals as discussed in Section 6. The test statistics corresponding to each distance metric are normally distributed, and so with 30 intervals, one can compute p-values with enough accuracy.

We performed extensive experiments to compare the distance metrics for several cases including for different subsets of iid hosts. We also conducted positive and negative control experiments where the cluster distributions were generated and known a priori. The following conclusions were found to be consistent across all the cases considered.

We find that Kolmogorov-Smirnov and Kuiper are not very sensitive in terms of distance as they are only concerned with extreme bounds. Cramer-von Mises and Anderson-Darling revealed that they are good candidates of clustering, but Anderson-Darling has high time and memory complexity. Moreover, Cramer-von Mises is advantageous for right-skewed distributions [11]. Thus we used the Cramer-von Mises as the distance metric when clustering.

We also conducted a sensitivity analysis with respect to the 30-sample threshold. We observed that using different samples size in the range of 20 to 25 does not effect the result of the distance metric nor clustering (specifically in terms of the fitted distributions).

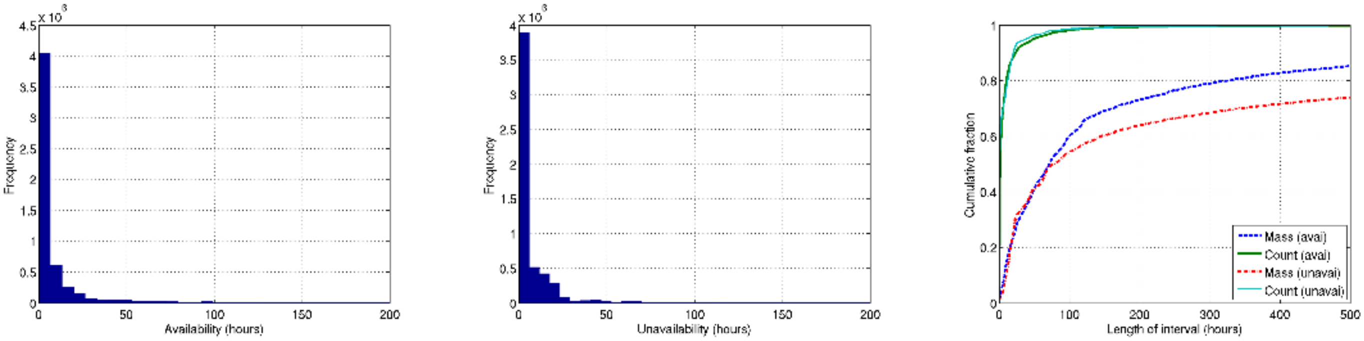
7.0.3 Cluster Results and Justification

We cluster hosts first by their distribution of availability. For all hosts in each of these clusters, we model the distribution of availability. Then we model the distribution of unavailability for all hosts in the cluster, defined by the availability distribution. The validity of this approach and the resulting models will be shown in Sections 7 and 8.

Alternatively, we tried to cluster hosts by both availability and unavailability distributions, using the vector

Test	Runs std		Runs up/down		Kendall		All		
	avai	*unavail*	*avai*	*unavail*	*avai*	*unavai*	*avai*	*unavai*	**Total**
# of hosts	101649	121385	144656	129267	101462	113973	65683	69385	**35686**
Fraction	0.602	0.719	0.857	0.766	0.601	0.675	0.389	0.411	**0.211**

TABLE 1
Result of Randomness Tests



(a) Availability Histogram

(b) Unavailability Histogram

(c) Mass-count

Fig. 3. Distribution of Availability and Unavailability Intervals

$< d_a, d_u >$ as the distance metric; d_a is the distance between two availability distributions, and d_u is the distance between the two corresponding unavailability distributions . However, this resulted in only two clusters, one of which was much larger than the other. Combining availability and unavailability greatly reduced the sensitively of our clustering method, and accuracy of our models. This can be explained by the fact that the distribution of unavailability across hosts does not differ as much as the distribution of availability. Thus, clustering by availability should be the first priority.

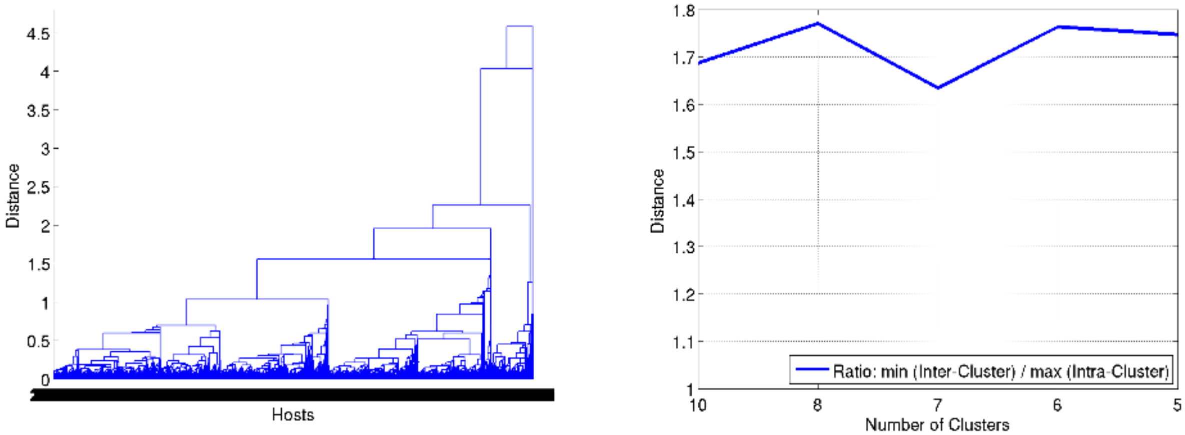
We found that the optimal number clusters was 6. We justify this number of clusters through several means. First, we observed the dendrogram as a result of hierarchical clustering for a random subset of hosts (due to memory consumption and scaling issues). As shown in Figure 4(a) the tree is an unbalanced tree where the height of tree shows the distance between different (sub)clusters. The good separation of hosts in this figure confirmed the advantage of Cramer-von Mises as the distance metric (in comparison to the result with other metrics). The number of distinct groups in this dendrogram where the distance threshold is set to one reveals that number of clusters should be between 5 to 10.

Second, using the result of hierarchical clustering as a bootstrap, we run k-means clustering *for all iid hosts* and then compute the within-group and between-group distances for various values of k . We utilized the Dunn index [25] to choose the best cluster size. This index could be simply described as the ratio of minimum inter-cluster over maximum intra-cluster distances. We observe that this ratio is maximum and about the same for eight and six clusters (see Figure 4(b)). However, cluster size of six is more homogeneous in terms of inter-cluster distance. Specifically, for k of 6, the inter-cluster distance between all clusters was maximized relative to other values of k. Also, for k of 6, the distance among all clusters is roughly equal (about 5). Third, we plotted the EDF corresponding to each cluster for a range of k. In this way, we can observe convergence or divergence of clusters during their formation.

Fourth, we plotted the EDF corresponding to each cluster. Figure 5 shows good separation of these plots. Based on the clusters defined by the distribution of availability, we show the corresponding EDF for unavailability in Figure 6. Note that the scale for unavailability in Figure 6 is about two orders of magnitude smaller than availability. In comparison, we found that the distributions of unavailability are heavier-tailed and more homogeneous than the distributions of availability. This is why we cluster first by availability.

7.1 Parameter Estimation for Clusters

After cluster discovery, we conduct parameter fitting for various distributions, including the exponential, hyperexponential, Weibull, Log-normal, Gamma, and Pareto. Parameter fitting was conducted using maximum likelihood estimation (MLE). Intuitively, MLE maximizes the log likelihood function that the samples resulted from a distribution with certain parameters. For hyperexponential fitting, we used expectation maximization (EM) which is based on MLE but can not guarantee the optimal answer. We used the EMpht package [15] with the same techniques described in [27]. We decided against using moment matching as it is sensitive to outliers [12] and the overall distributions are heavily right-skewed.



(a) Dendrogram of hierarchical clustering for a random subset of hosts

(b) Comparison of distances in clusters

Fig. 4. Cluster Comparison

We measured the goodness of fit (GOF) of the resulting distributions using standard probability-probability (PP) plots as a visual method and also quantitative metrics, i.e., Kolmogorov-Smirnov (KS) and Anderson-Darling (AD) tests. We show the graphical results of the best fits in Figure 7 for availability and unavailability.

For availability, we see that in most cases the Gamma distribution is a good fit for availability. Also, the exponential distribution has some close fits, especially in cluster 4. These result allows an approximation that would result in a simple analytical model. It is obvious from the GOF tests that the Pareto is completely far from our underlying distribution. So we do not have a heavy-tailed distribution. However, as in some plots, the Log-normal and Weibull have a close match so some cluster distributions are likely to be long-tailed.

We also used the same technique described in [27] to fit the hyper-exponential distribution for availability each cluster. Due to space limitations, we are not able to show the outcome, but the results revealed that the best fit for the one long-tailed cluster (i.e., cluster 3) is a 3-phase hyper-exponential distribution.

For unavailability, the hyper-exponential is a good fit in all cases and is better than the Log-Normal with exception of cluster 3. A three-phase hyper-exponential with several degrees of freedom was needed to create an accurate model, given fluctuations of the CDF for unavailability shown in Figure 6.

To be more quantitative, we also report the p-values of two goodness-of-fit tests. We randomly select a sub-sample of 30 of each data set and compute the p-values iteratively for 1000 times and finally obtain the average p-value. This method is similar to the one used by the authors in [27], [34], and was suggested to us by a statistician. Moreover, we observed that that the Coefficient of Variation of the p-value is less than 0.50. So the average value is a representative estimate.

The results of GOF tests are listed in Tables 2 and 3, where in the each row the best fit is highlighted. These quantitative results strongly confirm the graphical result of the PP-plots. (In the PP-plots, the closer the plots are to the line $y = x$, the better the fit.) With respect to availability, we find that the best-fitting distributions of each cluster differs significantly from the overall distribution over all iid hosts. Thus, clustering was essential for accurate availability modelling.

To be more precise, Table 4 lists some properties of iid hosts and clusters as well. The main point is the existing heterogeneity in the clusters in terms of the number of members and percentage of total availability contributed. The biggest cluster (i.e, cluster 3) that includes about 65% of all iid hosts, only contributes about 34% of the total availability. Cluster 2 that has about 11% of all iid hosts contributes 25% of total availability.

One question regarding the availability modelling is why are the five clusters that follow the Gamma distributions are not in one cluster. The answer is in the fifth column of Table 4 where the shape and scale parameters of fitting are reported. These values revealed that all five clusters have almost same shape (see Figure 5) but have very different scales.

Another question is whether or not we are able to model all hosts as a single distribution for availability, such as Gamma distribution with fixed parameters. The answer is no; the shape and scale parameters for the Gamma distribution fitting for all iid hosts are 0.2572 and 43.1661, respectively, which does not reflect the properties of different distributions listed in Table 4.

However, for unavailability distribution we could give completely different answers for above questions. As we mentioned before, the unavailability behavior of all iid hosts almost follow the same distribution. We are able to model unavailability with the hyper-exponential distribution with two different sets of parameters. For instance, as shown in the last column of Table 4, cluster 3 could form one group, and the remaining clusters could form another.

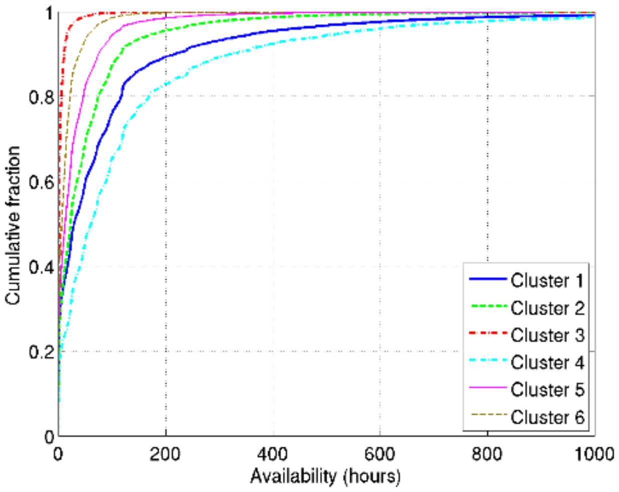


Fig. 5. EDF of availability clusters

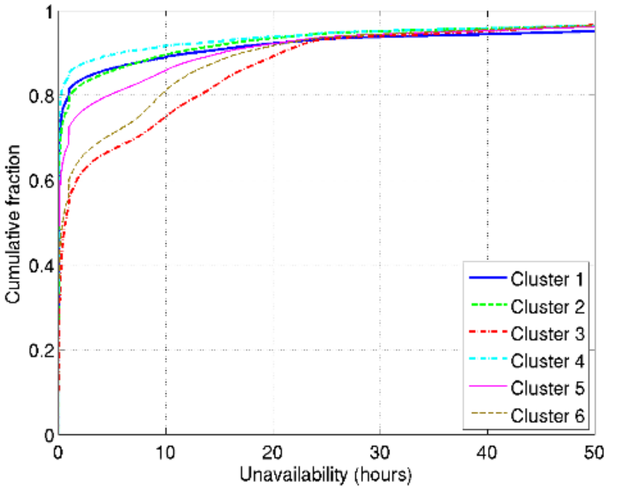


Fig. 6. EDF of unavailability corresponding to availability clusters

In summary, based on the graphical and quantitative p-value results, the Gamma distribution is a good fit for availability, and the hyper-exponential is a good fit for unavailability, for all clusters. So we use a single distribution for availability, and another for unavailability, but with different parameters (at least different scales) to model the different hosts. It is worth nothing that the Gamma and hyper-exponential distributions have very interesting properties in terms of their flexibility and generalization, and can be easily used in an analytical Markov models [12].

7.2 Significance of the Clustering Criteria

Our results in the previous section beg the following question: could the same clusters have been found using some other static criteria? In this section we consider cluster formation by static criteria, namely host venue, time zone, and clock rates. (Note that a small fraction of hosts did not have this information specified, and were thus excluded from this analysis.)

We define host venue to be whether a host is used at home, school, or work. This is specified by the user to the BOINC client (though this is not required). Roughly, 14,292 are at home, 3,211 are at work, and 434 are at school. If our clustering results correspond to those categories, we would expect the distribution of host venue across clusters to be even more skewed.

To measure this, we computed the expected number of hosts of a particular venue for each one of the six clusters identified in Section 7.0.3. The expected number is computed using the global percentage of home, work, and school hosts. The we counted the actual number of hosts of each venue in each cluster.

Figure 8(a) shows the expected number versus the actual number for each venue type over each cluster. The results for larger cluster sizes are more significant. If the clusters correspond to host venues, we would expect large deviations from the $y = x$ line. However, this is not the case as the expected and actual values are similar. Thus, the same clusters would not have resulted from the host venue.

We conducted the same comparison using host time zones. We counted the number of hosts in each cluster for each of the six largest time zones (in terms of hosts). These time zones corresponded to Central Europe (11,368 hosts), Eastern North America (6,359 hosts), Central North America (3,615 hosts), Western Europe (3,039 hosts), Western North America (2,726 hosts), and Eastern Asia (1,451 hosts).

We then compared this with the expected number, given the number of hosts in total for each of the six time zones. We find again that the corresponding points in Figure 8(a) are close the the line $y = x$. This indicates that the clusters are not a direct result of time zones alone.

We also investigated the relationship between CPU speeds (in FLOPS) and the clusters. Figure 8(b) shows a box-and-whisker plot of the CPU speeds for each cluster. The box represents the inter-quartile range. As this box for each cluster appears in similar ranges, we conclude that the clusters could not have been formed using CPU speeds alone, and that there is little correlation between CPU speeds and the length of intervals.

Nonetheless, one would like an intuition to explain why there was a formation of six clusters. We do not have a precise answer to this question. It could depend very much on the behavior and activities of the user.

8 Applications to Scheduling: the Resource Brokering Problem

Our goal here is two-fold. First, we show the utility of our models by applying them to a resource brokering problem. Second, we show the accuracy of our models

Data sets	Exponential		Pareto		Weibull		Log-Normal		Gamma	
	AD	*KS*	*AD*	*KS*	*AD*	*KS*	*AD*	*KS*	*AD*	*KS*
All iid hosts	0.008	0.000	0.052	0.015	**0.590**	**0.474**	0.543	0.393	0.474	0.402
Cluster 1	0.118	0.051	0.044	0.011	0.484	0.266	0.311	0.136	**0.554**	**0.373**
Cluster 2	0.185	0.086	0.020	0.004	0.480	0.267	0.316	0.143	**0.553**	**0.383**
Cluster 3	0.025	0.004	0.038	0.011	**0.574**	**0.454**	0.526	0.366	0.511	0.444
Cluster 4	0.250	0.153	0.004	0.000	0.464	0.226	0.223	0.081	**0.521**	**0.295**
Cluster 5	0.157	0.066	0.020	0.004	0.518	0.321	0.357	0.192	**0.560**	**0.403**
Cluster 6	0.161	0.068	0.017	0.003	0.530	0.347	0.376	0.204	**0.583**	**0.437**

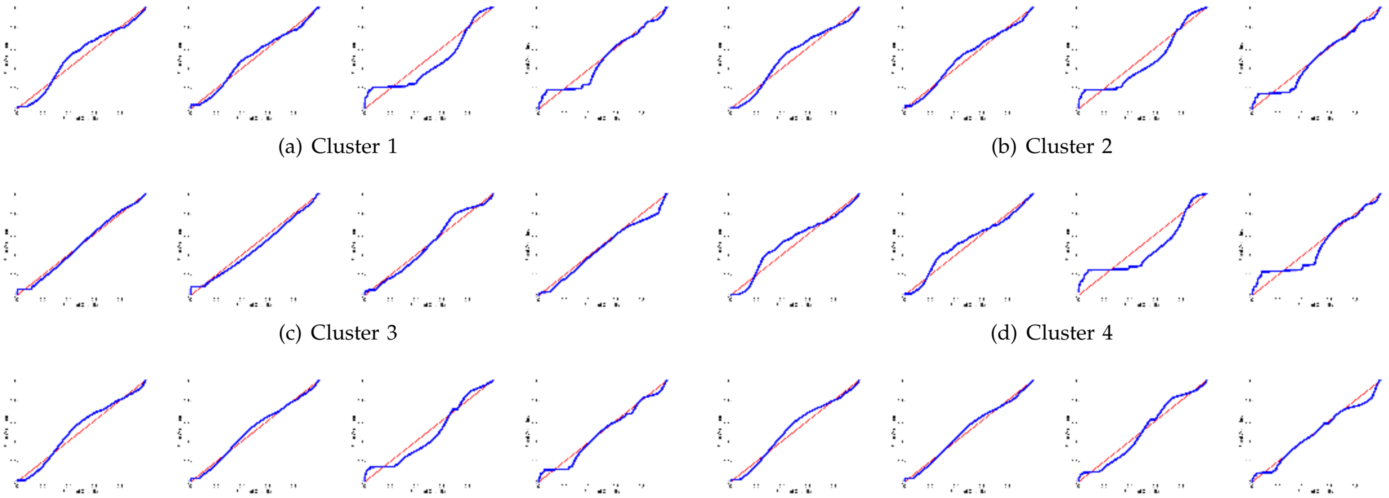
TABLE 2
P-value results from GOF tests for availability of all iid hosts and six clusters

Data sets	Hyper-exp.		Pareto		Weibull		Log-Normal		Gamma	
	AD	*KS*	*AD*	*KS*	*AD*	*KS*	*AD*	*KS*	*AD*	*KS*
All iid hosts	0.377	0.286	0.029	0.008	0.463	0.304	**0.514**	**0.322**	0.318	0.244
Cluster 1	**0.410**	**0.226**	0.006	0.001	0.119	0.035	0.246	0.123	0.032	0.008
Cluster 2	**0.450**	**0.296**	0.010	0.002	0.179	0.062	0.311	0.177	0.049	0.016
Cluster 3	0.459	0.342	0.026	0.006	0.497	0.313	**0.539**	**0.342**	0.370	288
Cluster 4	**0.362**	**0.142**	0.002	0.000	0.082	0.020	0.188	0.079	0.018	0.004
Cluster 5	**0.495**	**0.380**	0.019	0.005	0.278	0.150	0.400	0.243	0.106	0.049
Cluster 6	**0.468**	**0.358**	0.027	0.006	0.448	0.305	0.479	0.289	0.286	0.214

TABLE 3
P-value results from GOF tests for unavailability of all iid hosts and six clusters

Clusters	# of hosts	% of total avail.	Best fit (A)	Parameters		Best fit (U)	Parameters	
				shape	*scale*		P	μ
All iid hosts	35686	1.0	Weibull	0.393	2.964	Log-Normal	2.844	-0.586
Cluster 1	1516	0.13	Gamma	0.289	311.711	Hyper-exponential	0.141 0.683 0.176	0.008 37.816 1.013
Cluster 2	3863	0.25	Gamma	0.340	152.216	Hyper-exponential	0.142 0.660 0.198	0.013 30.636 0.777
Cluster 3	23494	0.34	Weibull	0.431	1.682	Hyper-exponential	0.398 0.305 0.298	0.031 11.566 1.322
Cluster 4	137	0.01	Gamma	0.357	371.622	Hyper-exponential	0.106 0.743 0.151	0.009 45.241 0.961
Cluster 5	3328	0.16	Gamma	0.342	89.223	Hyper-exponential	0.179 0.566 0.255	0.016 27.587 0.536
Cluster 6	3348	0.11	Gamma	0.357	43.652	Hyper-exponential	0.338 0.390 0.272	0.029 30.121 1.069

TABLE 4
Properties of clusters and iid hosts



(a) Cluster 1

(b) Cluster 2

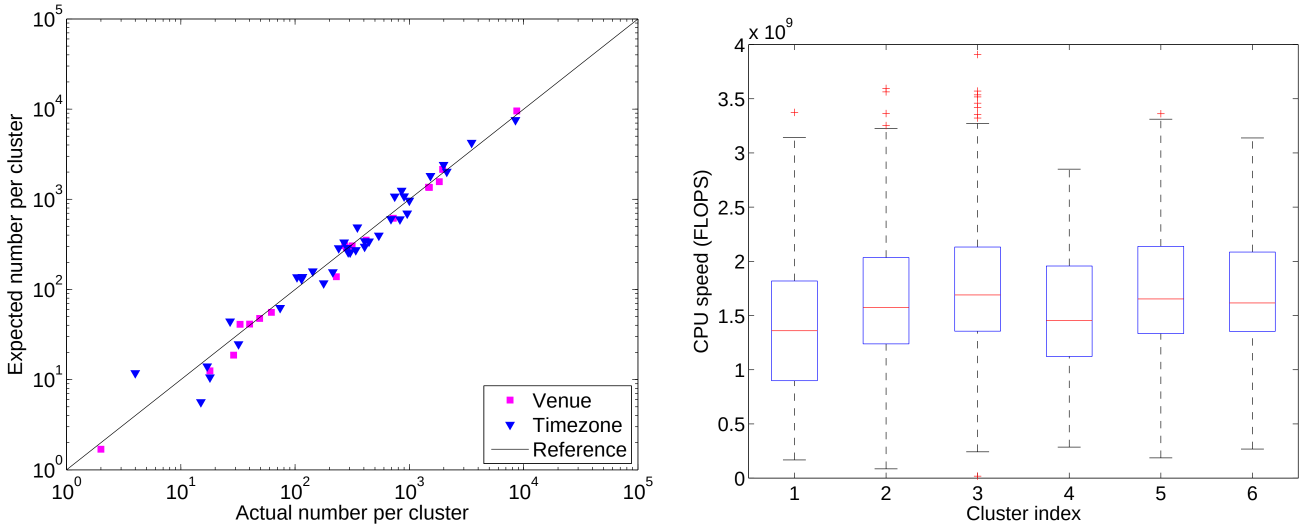
(c) Cluster 3

(d) Cluster 4

(e) Cluster 5

(f) Cluster 6

Fig. 7. PP-plots for all clusters. Four plots in row (left to right) per cluster correspond to Weibull/A, Gamma/A, Log-normal/U, Hyper-exponential/U. In each plot, X-axis: empirical quantiles, and Y-axis: fitted quantiles.



(a) By venue or time zone

(b) By CPU speed

Fig. 8. Clustering by other criteria

in the context of this scheduling problem. We do so at the local level by evaluating the model accuracy within a single cluster. Then we do so at the global level by evaluating the model accuracy over all clusters.

In the resource brokering problem, a **broker** must route a series of incoming **jobs** to a set of **schedulers**, one for each of $\boldsymbol{n}$ **clusters**. Each cluster contains $\boldsymbol{M_i}$ **nodes** (see Figure 9) where $1 \leq i \leq n$). The mean service rate of all nodes in cluster i is $\boldsymbol{\mu_i}$. The job submitter generates a workload of jobs with an arrival rate of $\boldsymbol{\lambda}$. Each job consists of $\boldsymbol{W}$ time units of work, contained in a set of **tasks**. We use the terms job and **bag of tasks (BOT)** interchangeably. A job is routed to one specific cluster versus being split among several. There is a work queue in the broker, the scheduler, and individual nodes.

When routing incoming jobs, the goal of the broker is to minimize the average **job completion time**, which is the time between a job is first submitted until it is completed entirely. The broker partitions the incoming sequence of jobs into n subsequences with rates λ_j where $\sum_{j=1}^{n} \lambda_j = \lambda$. λ_j is the incoming arrival rate of jobs routed to a particular scheduler. The scheduler in turn dispatches tasks on nodes within a cluster in FIFO order.

After pulling tasks from the scheduler, each node runs the tasks one by one and sends the results to the scheduler, and finally to the broker. In the case of node unavailability during task execution, we assume checkpointing so that the task is started from where it left off when the node becomes available again. To focus on availability modelling, we assume the nodes are homogeneous in terms of computing speed[1].

.

The resource brokering problem occurs in many real contexts. For example, in the World Community Grid, jobs from several different applications must be routed to a set of BOINC schedulers, each of which oversees a set of nodes [33]. In the EDGeS system that interconnects desktop Grids with traditional Grids, a centralized broker is used to route desktop grid jobs to one of several sub-Grids in EGEE [17]. In Condor[26], a broker must route jobs to several Condor pools in a flock.

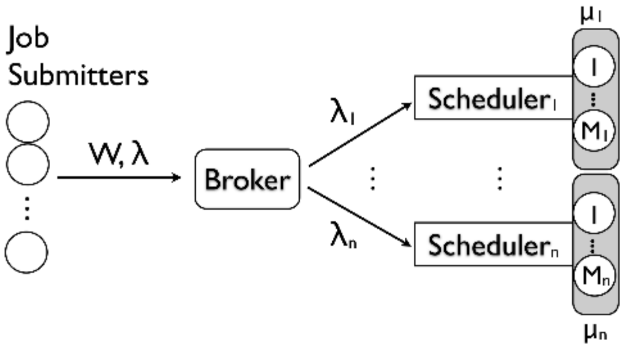


Fig. 9. Resource brokering diagram

8.1 Local Model

In this section, we evaluate the accuracy of the discovered models with respect to the measurement traces of an individual cluster. This corresponds to a resource broker with a low workload such that only one cluster is used at a time.

To do this, we need a reference model , and in [22], Kleinrock et. al. proposed a probability distribution function (PDF) of job completion time for a volatile distributed system where we have the availability and unavailability distributions. They used an Brownian motion approximation to model the system under study.

[1] To consider machine speed, we could further subdivide the clusters by clock rates, which is independent of availability and has a bell-shaped distribution.

They found that the PDF of the time t for M processors to complete W time units of work is given as follows:

$$f(t) = \frac{W}{\sqrt{2\pi\sigma_b^2 t^3}} \exp\left[-\frac{(W-\bar{b}t)^2}{2\sigma_b^2 t}\right] \tag{1}$$

where

$$\bar{b} = \frac{t_a}{(t_a+t_u)} M \tag{2}$$

$$\sigma_b^2 = \frac{\sigma_a^2 t_u^2 + \sigma_u^2 t_a^2}{(t_a+t_u)^3} M \tag{3}$$

The mean and variance are as follows:

$$\bar{f} = \frac{W}{\bar{b}} = \frac{W}{M}\frac{(t_a+t_u)}{t_a} \tag{4}$$

$$\sigma_f^2 = \frac{W}{\bar{b}}\frac{\sigma_b^2}{\bar{b}^2} \tag{5}$$

Moreover, t_a, t_u, σ_a^2 and σ_u^2 are the mean and the variance of availability and unavailability lengths, respectively.

The authors found that this model is accurate when we have a large workload with respect to the mean availability and unavailability lengths (i.e., $W \gg t_a + t_u$). Some approximations for the proposed distribution by a Normal or Log-normal distribution are presented in this work as well [22].

8.1.1 Model Inputs

To apply the model, we make the same assumptions as Kleinrock et al. described in [22]. (We loosen these assumptions later in Section 8.2.) First, we assume that the input workload is divisible, i.e., the workload can be divided in anyway among all available nodes. This is the case where we have large BOT's as the input workload, and we can keep all available nodes busy. Second, we use a constant amount of work per job (W) as utilized in [22][2] . Third, we use a job arrival rate that corresponds to a lightly loaded system such that there is no waiting time in the system queues. Jobs are generated according to a Poisson process with a low arrival rate that avoids waiting time in the queues (i.e., $rate = 10^{-10}$ per hour).

We simulate the behavior of a given node either using our fitted distributions or using the traces directly. In the case of model simulation, each node generates availability and unavailability intervals independently, based on the proposed distributions (see Table 4). However, for trace simulation, we do not have enough traces to run a long workload. To deal with this problem, we concatenate different traces randomly chosen from a single cluster to increase the total length. When a node finishes running jobs over one trace, we randomly select another trace from the same cluster and continue running. This process keeps the distribution and the mean value unchanged, but slightly changes the variance.

We must consider that nodes have different lifetimes in the system, which in turn effects the total number of active nodes in the system at any given time. An active node is defined to be a node that is registered and participating in the system, though is not necessarily available. We compute an average that gives the total number of active nodes in the cluster i at any given time as follows:

$$N_i = \frac{\text{Total lifetime of nodes}}{\text{Mean lifetime of nodes} \times M_i} \tag{6}$$

where M_i and N_i are the total and mean number of nodes in the cluster i respectively. Note that N_i underestimates the mean number of nodes in the system as we assume the beginning and end of the trace gives the lifetime.

8.1.2 Experimental Setup

In order to validate the discovered models, we implemented a discrete event-driven simulator. This simulator is developed using the Objective Modular Network Testbed in C++ (OMNeT++) simulation environment, which is an open-source, component-based and modular simulation framework [31].

The simulator uses the model or the traces to run the input workload.If a node is unavailable at the time of receiving a new task, the task will be inserted to the local queue of the node to be consistent with the Kleinrock model [22].

For each simulation experiment, statistics were gathered for a total number of 10,000 jobs. The first 1000 jobs during the *warm-up* phase were ignored to avoid bias before the system reached steady-state. The last 1000 jobs were ignored during the *drain* phase to allow previous jobs to run to completion. We have used the *batch means method* in our simulation experiments, where the jobs are split into many batches and statistics are accumulated for these batches. Since each sample in the batch means method is an average over many of the original samples, the variance among batches is greatly reduced, which in turn reduces the standard deviation of the measurements. This leads to better confidence in our estimates of the mean. In our experiments, the Coefficient of Variation of the results is very low (i.e., $CV < 0.05$).

8.1.3 Evaluation

Figure 10 shows the job completion time densities for each of the six clusters when we used a model or trace of availability and unavailability for the simulation of each node. Also, the third plot in each figure shows the Kleinrock model for each cluster based on the discovered model. The discovered model matches the Kleinrock model with a high degree of accuracy. Moreover, the results of the trace simulations are close. Cluster 3 has the greatest error because in this cluster, we have the shortest availability mean among all clusters, and the biggest intra-cluster distance. The mean job completion

[2] We also test uniform and exponential distributions for W and obtained similar results.

time of our derived model and Kleinrock model overestimate slightly the mean from the trace; so the models provide tight upper bounds on completion time.

Additionally, we report the mean job completion time in Table 5. The number of nodes and the mean availability and unavailability are listed in the table for all clusters. The last column in the table is the relative error of the trace versus the model. In general, the error rates are below 3%, with the exception of cluster 3 with an error of 13.25%. We show in next section that the local models have sufficient accuracy and provide a good fit when applied and evaluated globally.

8.2 Global Model

In this part, we want to evaluate the whole system and validate the discovered models versus traces for realistic scenarios.

8.2.1 Brokering Scheme

One of the important parts of the system under study is the brokering algorithm. If we consider each cluster as one "server" with a given service rate while the scheduler serves as an input queue and the broker as a router, the problem will simplify to the routing in parallel queues [4] (see Figure 9).

We consider three brokering algorithms, namely **Bernoulli**, **Billiard** and **Random**, which differ in two ways. First, they differ in how they compute the routing probabilities P_i corresponding to each cluster. Second, they differ in how they choose the sequence of jobs sent to each cluster.

The Random algorithm is the simplest way for routing. In this case, the routing probability of cluster i is $P_i = 1/n$, and the broker randomly chooses one cluster out of n clusters.

By contrast, the Bernoulli and Billiard algorithms use availability and unavailability models of nodes to determine the routing probabilities. Using the Bernoulli algorithm, the broker only uses routing probabilities without any special sequencing of jobs sent to clusters. In this sense, the Bernoulli algorithm is memoryless as it does not take into account what jobs have been sent to which clusters. The Billiard algorithm takes into account the past sequence of routing with a very limited cost.

The optimal routing probability P_i for Bernoulli and Billiards is computed as follows [5]. We define the service rate of each cluster as the reciprocal values of mean job completion time for a given workload as follows:

$$\mu_i = \left(\frac{\overline{W}}{N_i} \frac{t_{ai} + t_{ui}}{t_{ai}} \right)^{-1} \tag{7}$$

where $\overline{W}$ is the mean amount of work per job. The problem of how to find the optimal arrival rates in order to minimize the response time can be solved through the following mathematical programs, called *social optimization*:

$$\min_{\lambda_1,\ldots,\lambda_n} \sum_{j=1}^{n} \frac{\lambda_j}{\mu_j - \lambda_j} \tag{8}$$

where we have $\sum_{j=1}^{n} \lambda_j = \lambda$ and $0 \leq \lambda_j < \mu_j$ for all $j \in 1, 2, ..., n$. It should be noted that this optimization was proposed when the service rates follow the exponential distribution, However we apply it for the case where the distribution is general. Bell and Stidham [5] presented an optimal solution to find the first k clusters to use with the following condition:

$$r_k < \lambda \leq r_{k+1} \text{ where } r_k = \sum_{i=1}^{k} (\mu_i - \sqrt{\mu_k \mu_i}) \tag{9}$$

They assume that $\mu_1 \geq \mu_2 \geq ... \geq \mu_n > 0$, $r_1 = 0$ and $r_n = \sum_{i=1}^{n} \mu_i$. Consequently, the optimal arrival rate for k clusters is provided by:

$$\lambda_i^k = \mu_i - \frac{\sqrt{\mu_i} \left(\sum_{j=1}^{k} \mu_j - \lambda \right)}{\sum_{j=1}^{k} \sqrt{\mu_j}} \quad \text{for } 1 \leq i \leq k \tag{10}$$

It should be noted that arrival rate for the other clusters beyond k would be zero ($\lambda_i = 0, k < i \leq n$). Finally, the routing probabilities can be calculated as $P_i = \lambda_i^k / \lambda$.

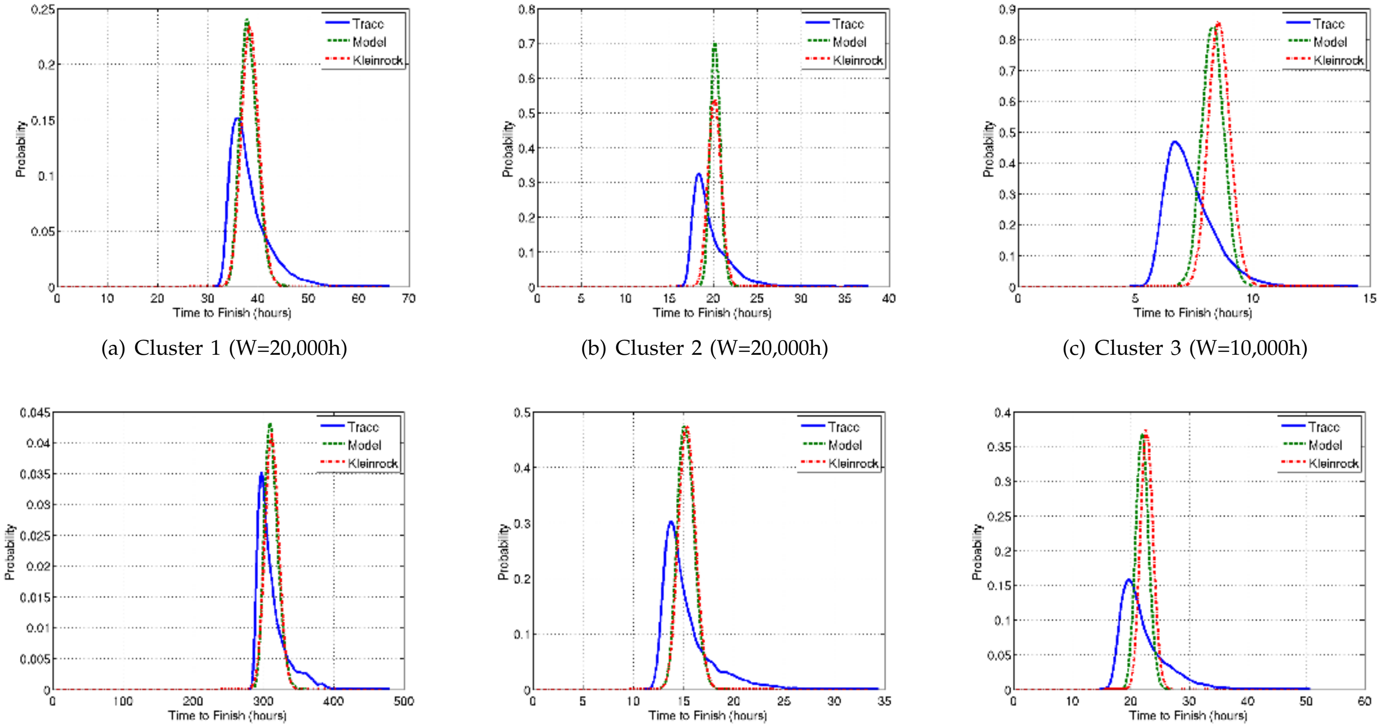
The Bernoulli and Billiard algorithms use same routing probabilities and differ only in the sequences of jobs sent to each cluster. Billiard is a generalized form of round-robin, and it takes into account the sequence of routing called the billiard sequence [19], determined as follows.

Suppose that a billiard ball bounces in an n-dimensional cube where each side and opposite side are assigned by an integer value in range of $\{1, 2, ..., n\}$. Then, the billiard sequence is generated by the series of integer values which show the sides hit by the ball when shot. This sequence is deterministic, and different from the sequence of Bernoulli scheme, which is completely random.

In [19], the authors proposed a method to implement this scheme and generate the billiard sequence as follows:

$$i_b = \min_{\forall i} \left\{ \frac{X_i + K_i}{P_i} \right\} \tag{11}$$

where i_b is the "winning" cluster, and K_i and X_i are vectors of integers with size n. K_i keeps track the number of jobs that have been sent to the cluster i. X_i reflects which cluster is fastest, and is set to one for the fastest cluster and zero for all other clusters [4]. Thus K_i has to be initialized to zero, and after finding the target cluster, it must be updated as $K_{i_b} = K_{i_b} + 1$. P_i is the fraction of jobs that sent to the cluster i and is the same as in the Bernoulli scheme. It has been shown in [4] that the proposed method will provide the optimal response time when all service times follow the exponential distribution.



(a) Cluster 1 (W=20,000h)

(b) Cluster 2 (W=20,000h)

(c) Cluster 3 (W=10,000h)

(d) Cluster 4 (W=20,000h)

(e) Cluster 5 (W=10,000h)

(f) Cluster 6 (W=10,000h)

Fig. 10. Job completion time densities for each cluster for proposed model, trace and Kleinrock model

Clusters	N	Parameters (hrs)		Mean job completion time (hrs)			Relative error(%)
		t_a	t_u	*Trace*	*Model*	*Kleinrock*	
Cluster 1	622	90.22	17.42	38.15	38.18	38.38	0.09
Cluster 2	1211	51.73	11.40	19.62	20.22	20.15	3.0
Cluster 3	4491	4.62	13.19	7.33	8.30	8.58	13.25
Cluster 4	70	132.66	12.15	311.27	311.46	311.89	0.06
Cluster 5	906	30.54	11.93	15.23	15.24	15.35	0.08
Cluster 6	774	15.58	11.87	22.06	21.99	22.76	0.32

TABLE 5
Comparison of clusters to evaluate the accuracy of the discovered models

8.2.2 Model Inputs

The workload model for evaluation of the whole system is obtained from the Grid Workload Archive [20]. We used the BOT model [21] which adopted Weibull and Gamma distributions for the inter-arrival rate of jobs and the BOT size (i.e., the number of tasks in each BOT), respectively. Moreover, we utilized the Normal distribution for task runtimes within a BOT, using parameters of real BOINC projects [16], [21]. The parameters of the distributions are listed in Table 6. Since we need to evaluate all clusters in the system, we chose an input rate such that all clusters be in used (i.e., $k = n$). In the other words, the system would be in the high load, so the job completion time includes the waiting time in the queues as well.

Input Parameters	Distribution
BOT runtime 1	Normal($\mu = 2.1, \sigma = 1.4$)
BOT runtime 2	Normal($\mu = 1.2, \sigma = 0.9$)
BOT size	Gamma($\alpha = 5, \beta = 200$)
Arrival rate	Weibull($\alpha = variable, \beta = 1.5$)

TABLE 6
Input parameters for the global model

8.2.3 Experimental Setup

We used the same experimental setup as before, and only the scheduling mechanism of clusters has been changed. In the new setup, the cluster scheduler only dispatches the jobs to the available nodes in the cluster and not all nodes to be consistent with the model used in [4]. So only the queue of the scheduler (versus the nodes) is used in practice, and we only deal with one queue for each cluster. For each simulation experiment, a total number of 250,000 jobs were run in ten batches, each with 25,000 jobs. Statistics were gathered for the eight middle batches when the system reached steady state.

In our experiments, the Coefficient of Variation of the simulation results is very low (i.e., $CV < 0.05$).

8.2.4 Evaluation

The results of simulation for model and trace for the system with $\overline{W} = 2100$ and $\overline{W} = 1200$ hours are depicted in Figures 11(a) and 11(b). Here, W is the product of the BOT size and task runtime. The mean job completion times are plotted against the job generation rate for the three different brokering schemes, respectively. The bars reflect the 95% confidence intervals, and the distribution of job completion times.

First, the figures reveal that the result of model and trace are close with a good degree of accuracy where the maximum relative error is less than 10%. Secondly, as we expected, the performance of the Random broker is the worst. Thirdly, the Billiard scheme performs better than Bernoulli; while both algorithms used the same routing probabilities, they dispatch jobs to clusters in different orders, and this in turn explains the better performance of the Billiard approach. Billiard outperformed Bernoulli with factor of 10 in terms of mean job completion time. This reveals the importance of sequencing in the broker.

For a broader comparison, Figure 11(c) shows the CDF of job completion times corresponding to trace and model simulations for all three brokering schemes and the arrival rate of 1.65 jobs per hour. The distribution of completion times from model and trace are close to each other.

These results confirm and reinforce the theoretical work in [4], which showed that Billiard provides the optimal response time when all service times follow the exponential distribution. Interestingly, the Billiard algorithm still achieves significant gains relative to other algorithms for heavier-tailed distributions based on our traces.

9 Conclusions

We considered the problem of discovering availability models for host subsets from a large distributed system. Our specific contributions were the following:

- *With respect to methodology.*
  - We detected and determined the cause of outliers in the measurements. We minimize their effect during the preprocessing phase of the measurements. This is important for others to use the trace data set effectively.
  - We described a new method for partitioning hosts into subsets that facilitates the design of stochastic scheduling algorithms. We use randomness tests to identify hosts with iid availability.
  - For iid hosts, we use clustering based on distance metrics that measure the difference between two probability distributions. We find that the Cramer-von Mises metric is the most sensitive and computationally efficient compared to others.
  - In the process, we describe how we deal with with large and variable sample sizes and the hyper-sensitivity of statistical tests by taking a fixed number (30) of random samples from each host.
- *With respect to modelling.* We apply the methodology to the one of the largest Internet-distributed platforms on the planet, namely SETI@home.
  - We find that about 21% of hosts have truly random availability intervals.
  - These iid hosts have can be separated into 6 groups, based on their distributions of availability and unavailability. Availability and unavailability in each group can be modelled accurately with a Gamma and hyper-exponential distribution respectively with different scale parameters. This is useful as these distributions can be used easily to construct analytical Markov models.
  - The discovered distributions from the same family differ greatly by scale, which explains their separation after clustering. The distribution for all hosts with iid availability also differs significantly in scale from any of the distributions corresponding to each cluster.
- *With respect to scheduling.* We show the utility and accuracy of our models in the context of a resource brokering problem.
  - We find that analytical results based on our models match theoretical results, and results from trace-driven simulation at local and global levels.
  - We find that the Billiard brokering method outperforms Random and Bernoulli methods by an order of magnitude in practice, reinforcing theoretical results.

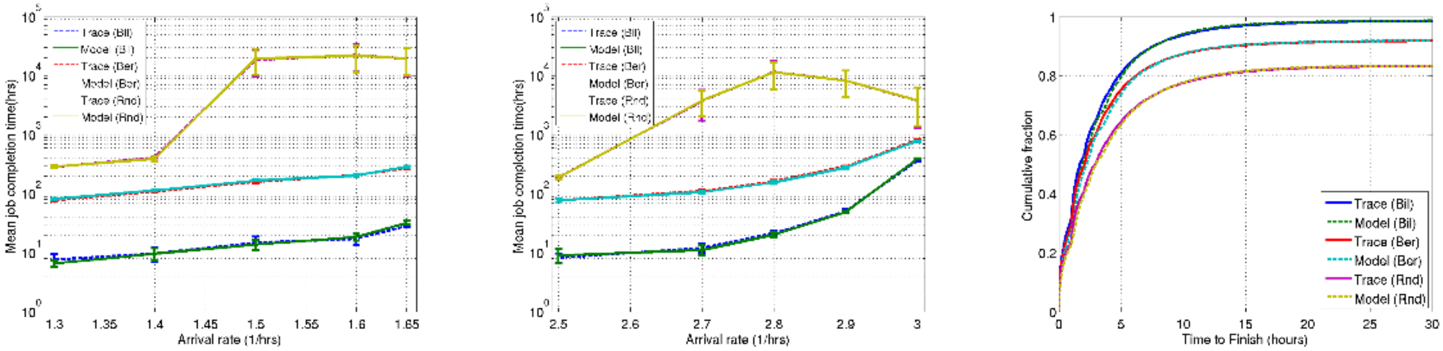
For future work, we would like apply our method to free resources in data centers or enterprise desktops, such as the un-dedicated Spot servers provided by Amazon's Elastic Compute Cloud or desktops at Microsoft Inc [8]. While the distribution of availability could change and become heavier tailed, we believe our clustering and scheduling methods are general enough for useful application.

10 Availability of Data and Tools

The SETI@home trace data and analysis tools have been released on the Failure Trace Archive, available online at: http://fta.inria.fr.

Acknowledgments

We thank Jonatha Anselmi, Bruno Gaujal, Eric Heien, Rich Ulrich for useful discussions. This work has been supported in part by the ANR project SPADES (08-ANR-SEGI-025), and the INRIA project CloudComputing@home.



(a) $\overline{W} = 2100h$

(b) $\overline{W} = 1200h$

(c) CDF for $\overline{W} = 2100h$ (rate=1.65)

Fig. 11. Mean job completion time of model and trace simulation for three different brokering schemes

References

[1] A. Acharya, G. Edjlali, and J. Saltz. The Utility of Exploiting Idle Workstations for Parallel Computation. In *Proceedings of the 1997 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 225–234, 1997.

[2] D. Anderson. Boinc: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, 2004.

[3] D. Anderson and G. Fedak. The Computational and Storage Potential of Volunteer Computing. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, 2006.

[4] J. Anselmi and B. Gaujal. On the price of anarchy and the optimal routing of parallel non-observable queues. Research Report 00457603, INRIA Rhone-Alpes, February 2010.

[5] C. Bell and S. Stidham. Individual versus social optimization in the allocation of customers to alternative servers. *Management Science*, 29:831–839, 1983.

[6] Anne Benoit, Yves Robert, Arnold L. Rosenberg, and Frédéric Vivien. Static strategies for worksharing with unrecoverable interruptions. In *IPDPS*, pages 1–12, 2009.

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

[8] 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.

[9] P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer Science-Business Media, Inc., 2002.

[10] G. Casella and R. Berger. *Statistical Inference*. Duxbury, 2002.

[11] G Cirrone et al. A goodness-of-fit statistical toolkit. *IEEE Transctions on Nuclear Science*, 51(5):2056–2063, 2004.

[12] Feitelson. D. *Workload Modeling for Computer Systems Performance Evaluation*. 2009.

[13] John R. Douceur. Is remote host availability governed by a universal law? *SIGMETRICS Performance Evaluation Review*, 31(3):25–29, 2003.

[14] Charles Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, pages 147–153, 2003.

[15] EMpht home page. http://home.imf.au.dk/asmus/pspapers.html.

[16] T. Estrada, K. Reed, and M. Taufer. Modeling job lifespan delays in volunteer computing projects. In *Proceedings of to the 9th IEEE International Symposium on Cluster Computing and Grid (CCGrid)*, 2009.

[17] Peter Kacsuk et al. Edges: Bridging egee to boinc and xtremweb. *Journal of Grid Computing*, 7(3):335–354, 2009.

[18] Eric Martin Heien, David P. Anderson, and Kenichi Hagihara. Computing low latency batches with unreliable workers in volunteer computing environments. *Journal of Grid Computing*, 7(4):501–518, 2009.

[19] A. Hordijk and D. A. van der Laan. Bounds for deterministic periodic routing sequences. *LNCS*, 2081:236–250, 2001.

[20] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D.H.J. Epema. The grid workloads archive. *Future Generation Computer Systems*, 2008.

[21] A. Iosup, O. Sonmez, S. Anoep, and D.H.J.Epema. The performance of bags-of-tasks in large-scale distributed computing systems. In *In the IEEE Symp. on High Performance Distributed Computing (HPDC08)*, Boston, US, 2008.

[22] L. Kleinrock and W. Korfhage. Collection unused processing capacity: An analysis of transient distributed systems. *IEEE TPDS*, 4(5):535–546, 1993.

[23] 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.

[24] Derrick Kondo, Artur Andrzejak, and David P. Anderson. On correlated availability in internet distributed systems. In *IEEE/ACM International Conference on Grid Computing (Grid)*, Tsukuba,Japan, 2008.

[25] Csaba Legany, Sandor Juhasz, and Attila Babos. Cluster validity measurement techniques. In *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, September 2006.

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

[27] D. Nurmi, J. Brevik, and R. Wolski. Modeling Machine Availability in Enterprise and Wide-area Distributed Computing Environments . Technical Report CS2003-28, Dept. of Computer Science and Engineering, University of California at Santa Barbara, 2003.

[28] S. Saroiu, P.K. Gummadi, and S.D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedinsg of MMCN*, January 2002.

[29] Jason D. Sonnek, Mukesh Nathan, Abhishek Chandra, and Jon B. Weissman. Reputation-based scheduling on unreliable distributed infrastructures. In *ICDCS*, page 30, 2006.

[30] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *ICM'06*, Rio de Janeiro, Brazil, 2006.

[31] Andras Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *International Conference on Simulation Tools and Techniques for Commuications, Networks and Systems*, Marseille, France, 2008.

[32] Ying Wang. Nonparametric tests for randomness. Research report, UIUC, May 2003.

[33] World community grid. http://www.worldcommunitygrid.org/.

[34] J. Wingstrom and H. Casanova. Statistical modeling of resource availability in desktop grids. Research Report RR-ICS2007-11-01, Universit of Hawaii, November 2007.

[35] J. Wingstrom and H. Casanova. Probabilistic allocation of tasks on desktop grids. In *IPDPS'08*, pages 1–8, 2008.