A Statistical Approach to Risk Mitigation in Computational Markets

Thomas Sandholm
KTH – Royal Institute of Technology
Center for Parallel Computers
SE-100 44 Stockholm, Sweden
sandholm@pdc.kth.se

Kevin Lai
Hewlett-Packard Laboratories
Information Dynamics Laboratory
Palo Alto, California 94304
kevin.lai@hp.com

ABSTRACT

We study stochastic models to mitigate the risk of poor Quality-of-Service (QoS) in computational markets. Consumers who purchase services expect both price and performance guarantees. They need to predict future demand to budget for sustained performance despite price fluctuations. Conversely, providers need to estimate demand to price future usage. The skewed and bursty nature of demand in large-scale computer networks challenges the common statistical assumptions of symmetry, independence, and stationarity. This discrepancy leads to underestimation of investment risk. We confirm this non-normal distribution behavior in our study of demand in computational markets.

The high agility of a dynamic resource market requires flexible, efficient, and adaptable predictions. Computational needs are typically expressed using performance levels, hence we estimate worst-case bounds of price distributions to mitigate the risk of missing execution deadlines.

To meet these needs, we use moving time windows of statistics to approximate price percentile functions. We use snapshots of summary statistics to calculate prediction intervals and estimate model uncertainty. Our approach is model-agnostic, distribution-free both in prices and prediction errors, and does not require extensive sampling nor manual parameter tuning. Our simulations and experiments show that a Chebyshev inequality model generates accurate predictions with minimal sample data requirements.

We also show that this approach mitigates the risk of dropped service level performance when selecting hosts to run a bag-of-task Grid application simulation in a live computational market cluster.

Categories and Subject Descriptors

D.4.8 [**Operating Systems**]: Performance—*Modeling and prediction*

HPDC'07, June 25–29, 2007, Monterey, California, USA.

General Terms

Management, Performance

Keywords

QoS, Service Level Management, Resource Allocation

1. INTRODUCTION

Large scale shared computational Grids allow more efficient usage of resources through statistical multiplexing. Economic allocation of resources in such systems provide a variety of benefits including allocating resources to users who benefit from them the most, encouraging organizations to share resources, and providing accountability [29, 15, 5, 31].

One critical issue for economic allocation systems is predictability. Users require the ability to predict future prices for resources so that they can plan their budgets. Without predictability, users will either over-spend, sacrificing future performance by prematurely spending the risk-hedging buffer, or over-save, sacrificing current performance. Both lead to dissatisfaction and instability. Moreover, the lack of accurate information precludes rational behavior, which would disrupt the operation of the many allocation mechanisms that depend on rational behavior.

There are three parts to predictability: the predictability provided by the allocation mechanism, the predictability of the users' behavior, and the predictability provided by statistical algorithms used to model the behavior. In this work we focus on the last part. The first part was investigated in [11] and examples of mechanisms addressing the second part include [7, 30].

The goal of this paper is to examine the degree to which future QoS levels, guarantees, and resource prices can be predicted from previous demand on a shared computing platform. Ideally, we would use the pricing data from a heavily used economic Grid system, but such systems have not yet been widely deployed. Instead, we examine PlanetLab[20], a widely-distributed, shared computing platform with a highly flexible and fluid allocation mechanism. The PlanetLab data set has the advantage of encompassing many users and hosts and having very little friction for allocation. However, PlanetLab does not use economic allocation, so we substitute usage as a proxy for pricing. Since many economic allocation mechanisms (e.g., Spawn[25], Popcorn[21], ICE[9], and Tycoon[16]) adjust prices in response to the demand, we believe that this approximation is appropriate.

We also analyze job traces from the Royal Institute of Technology in Stockholm (KTH), Ohio Supercomputing Center (OSC), and San Diego Supercomputer Center (SDSC) to contrast the PlanetLab load with deployments exhibiting more traditional high performance computing user behavior.

In these traces, we examine three key statistical metrics computed on the time-varying demand: distribution skewness, correlation over time, and volatility over time (heteroskedacity). Skewness captures whether most of the distribution's values are closer to the lower bound (right-skewed), upper bound (left-skewed), or neither (symmetric). Correlation over time measures the relative importance of old and recent samples in predicting future samples. Finally, heteroskedacity measures how variance changes over time. We find that that the traced demand distributions have significant skewness, long-term correlation, and changes in volatility over time.

Common modelling techniques fail under these conditions. We show using simulation that a predictor based on a Normal (Norm) Gaussian distribution model and a benchmark (Bench) predictor using the entire sample history both perform poorly with highly skewed data. In addition, high heteroskedacity reduces the accuracy of any predictions of the expected value of demand.

We develop a new predictor based on the Chebyshev (Cheb) model. It is distribution-free and provides worst-case scenario bounds independent of distribution symmetry. To handle time correlations, we sample statistics in running moments with exponential smoothing in different time windows (e.g., hour, day and week). This allows us to maintain less data as well as to adapt quickly to changing demand behavior.

To handle heteroskedacity, the Chebyshev-based predictor uses the running moments to predict demand bounds. We observe that users of a market-based system can substitute predictions of demand bounds for predictions of expected value. Market-based systems have mechanisms for insuring against future changes in prices (e.g., bidding more for resources). Demand bound predictions are sufficient for users to decide how much money to spend to mitigate risk. For example, a predicted price bound of [\$9, \$50] per GHz with an expected price of \$10 conveys that, despite low expected demand, there is significant risk of a high price and the user should spend more to mitigate risk.

We use a live computational market based on the Tycoon[16] resource allocator to compare a strategy using the Chebyshev-based predictor to the default strategy of using the spot market price when scheduling jobs. Using a varying load over time, we show that the predictor outperforms the spot market strategy while imposing a negligible (0.001%) computational and storage overhead.

The remainder of this paper is structured as follows. In Section 2 we outline the requirements of the prediction problem we are addressing. We analyze the demand traces by comparing the dynamics of the the different systems under investigation in Section 3. In Section 4, we describe in more detail the design and rationale of our prediction approach and models. In Section 5, we present and discuss the setup and results of the simulations and experiments. We review related work in Section 6 and conclude in Section 7.

2. REQUIREMENTS

2.1 User Requirements

Consumers in a computational market need guidance regarding how much to spend to meet their performance requirements and task deadlines. The quality of this guidance is critical to both individual and overall system performance. The best prediction system would recommend users to invest as little as possible while meeting their QoS requirements with as high a probability as possible. Robust statistical performance guarantees are an alternative to reservations with absolute guarantees, which are vulnerable to speculation and a variety of other gaming strategies [6].

In the discussion below, the *bid* is the amount the consumer pays a provider for a resource share over a given period of time. A higher bid obtains a greater share, or *QoS level*. The *guarantee* is the likelihood that the delivered QoS is greater than the consumer's defined value. The guarantee is not a contract, but rather the estimated probability that a QoS level can be delivered.

The prediction system answers the following questions for the user:

QUESTION 1. *How much should I bid to obtain an expected QoS level with a given guarantee?*

QUESTION 2. *What QoS level can I expect with a given guarantee if I spend a given amount?*

QUESTION 3. *What guarantee can I expect for a given QoS level if I spend a given amount?*

Exactly which question(s) a user asks varies from job to job and from user to user. We assume that users can ask any question at any time.

We further assume that providers allocate shares among consumers using the proportional share model as follows:

Definition 1. $Q = \frac{bid}{bid+Y}$

where Y denotes the demand of the resource modeled as a random variable with an arbitrary unknown distribution, and Q is the random variable representing the obtained performance when requesting performance qos. We define guarantee as the probability that the obtained performance is greater than the requested performance:

Definition 2. $g = P(Q > qos)$

PROPOSITION 1. *To get performance guarantee g, we need to spend $\frac{CDF^{-1}(g)qos}{1-qos}$, where CDF^{-1} is the percent point function or the inverse of the cumulative distribution function for the demand.*

PROOF. Substituting the share Q in Definition 2 with the right side of the equation in Definition 1 gives

$$g = P\left(\frac{bid}{bid+Y} > qos\right) = P\left(Y \leq \frac{bid}{qos} - bid\right) =$$

$$CDF\left(\frac{bid}{qos} - bid\right)$$

the inverse CDF of the demand distribution can then be expressed as:

$$CDF^{-1}(g) = \frac{bid}{qos} - bid$$

which after rearranging gives:

$$bid = \frac{CDF^{-1}(g)qos}{1 - qos}$$

□

This provides everything to answer Question 1, 2, 3. To obtain an expected QoS level with a given guarantee, a user should bid the following (Question 1):

$$bid = \left.\frac{CDF^{-1}(g)q}{1-q}\right|_{P(Q>q)=g}. \quad (1)$$

A user who bids a given amount and expects a given guarantee should expect the following QoS level (Question 2):

$$q = \left.\frac{bid}{bid + CDF^{-1}(g)}\right|_{P(Q>q)=g}. \quad (2)$$

A user who bids a given amount and expects a given QoS level should expect the following guarantee (Question 3):

$$g = CDF\left(\left.\frac{(1-q)bid}{q}\right)\right|_{P(Q>q)=g} \quad (3)$$

The main goal of our prediction method is to construct estimates of the CDF and CDF^{-1} (a.k.a. percent point function (PPF)) accurately and efficiently.

2.2 System Requirements

Two different approaches for estimating probability densities stand out: parameter-based and parameter-free estimation. In the parameter-free approach, one takes a random sample of data points, and smooths them to calculate the most likely real underlying distribution, e.g. using a least-squares algorithm. In the parameter-based approach, one assumes certain structural and behavioral characteristics about the real distribution and finds the parameters that best fit the measured data, e.g. using some maximum likelihood algorithm. In either case, sample measurements or data points are needed to calculate the density functions. Recording the history of demand in time-series streams for a large number of resources across a large number of computational nodes in real-time does not scale, in terms of state maintenance and distribution, and prediction model construction and execution. The scalability limitations force restrictions on the length of past and future prediction horizons and the number of resources that can be tracked. As a result, our goal is to use as few distribution summary data points as possible to make as flexible predictions as possible.

Studies of large-scale networked systems [19, 8] show that the underlying distribution of the demand is neither normal nor symmetric. Assuming that it is would result in underestimated risks, so accommodating bursty, skewed behavior is a necessity. Furthermore, we neither want to assume stationarity nor independence of the underlying distribution since consumers are interested in getting the most accurate estimate based on performing a task in the near future, and evaluate that option against waiting for better conditions.

There is a trade-off between performance and accuracy of the predictions, but there is also a similar trade-off between flexibility and evaluation capability. We would like to empower users to do rich customized predictions based on minimal summary statistics. They should be able to execute what-if scenario probes based on all three questions mentioned in Section 2.1. Different questions incur different prediction errors, which complicates the generic evaluation of model uncertainty and construction of prediction intervals.

3. DEMAND ANALYSIS

In this section we describe the data traces used in the simulations in Section 5. The load traces come from four shared compute clusters. The PlanetLab trace is available on-line at `http://comon.cs.princeton.edu/`. The San Diego Supercomputer Center, Ohio Supercomputing Center, and Stockholm Royal Institute of Technology traces are all available at `http://www.cs.huji.ac.il/labs/parallel/workload/`.

• **PlanetLab.** PlanetLab (PL) is a *planetary-scale*, distributed computing platform comprising approximately 726 machines at 354 sites in 25 countries, all running the same Linux based operating system and PlanetLab software stack, including a virtual machine monitor. The user community is predominantly computer science researchers performing large-scale networking algorithm and system experiments. Resources are allocated in a proportional share fashion, in virtual machine slices. The trace is from December 2005 to December 2006, and was obtained from PlanetLab CoMon node-level statistics. We calculate demand by aggregating the load value across all hosts for each 5-min snapshot interval available. This load measures the number of processes that are ready to run across all virtual machines. The load data was filtered to remove a periodic peak caused by synchronized rpm database updates across all slices appearing over a 30-min period every night, to get a more accurate representation of demand.

• **San Diego Supercomputer Center.** The San Diego Supercomputer (SDSC) trace was obtained from Dror Freitelson's parallel workloads archive [2]. It is the SDSC Blue Horizon workload from June 2000 to June 2001 (SDSC-BLUE-2000-3.1-cln.swf). The load is from the 144 node 8-way SMP crossbar Blue Horizon cluster using the LoadLeveler scheduler.

The demand is calculated in three steps. First, the CPU usage or load for a job is obtained as: $r_t(t_e - t_s)$, where r_t is the CPU time consumed by the job in percentage of the total run time, and t_e and t_s are the end time and start time in seconds since epoch respectively–all three values are available directly from the trace. Second, each job CPU usage is correlated to the time when it was submitted, thus effectively simulating that no queuing was done but all jobs could run with their actual load instantly. Finally, we aggregate the obtained CPU load value across all jobs running in every 5-min time interval. We did not analyze the utilization data directly because it could mask high demand under heavy load. The transformation also makes it comparable to proportional share systems such as PlanetLab and Tycoon, which are the primary focus of our work. Although this masks the needs of users who do not submit jobs when the queue wait-times are too long, we assume that these users would not spend money in an expensive computational market either. Consequently, we assume that these users do not contribute to demand.

• **Ohio Supercomputing Center.** The Ohio Supercomputing Center (OSC) trace was also obtained from the parallel workloads archive. It is the OSC Linux cluster workload from January 2001 to December 2001 (OSC-Clust-2000-2.swf). The site is a Linux cluster with 32 quad nodes and

Table 1: Central Moments of Traces (skewness > 2 is marked in bold to indicate a heavy tail)

	μ	$\frac{\sigma}{\mu}$	γ_1	γ_2
PL	3433	0.37	**4.06**	29.45
KTH	382	0.71	1.11	0.90
OSC	66	0.72	1.53	4.35
SDSC	3249	0.51	1.02	2.07

25 dual nodes with a total of 178 processors using the Maui scheduler. We perform the identical transformation from job workload to demand as with the SDSC data described above.

• **Royal Institute of Technology.** The Center for Parallel Computers at the Swedish Royal Institute of Technology (KTH) in Stockholm, provided us with the final workload trace. The trace is from a 100-node IBM SP2 cluster from October 1996 to September 1997 (KTH-SP2-1996-2.swf). Because CPU time was not recorded in the logs, the CPU load is set to the run time of the jobs. Apart from this the demand transformation is identical to the SDSC and OSC traces described above.

Next, we characterize the dynamics of the computational demand traces by examining the typical properties of time series distributions: symmetry, independence, and stationarity.

3.1 Distribution Symmetry

The first step towards characterizing the load and detecting anomalies is to look at the raw demand traces. Figure 1 shows that PlanetLab exhibits much thinner peaks that both appear and disappear more quickly than the peaks in the other traces. We attribute this behavior to the fact that PlanetLab jobs tend to be distributed systems or networking experiments that have lower resource requirement than scientific computing workloads.

Next, we measure the distribution symmetry of the demand for the different clusters in Figure 2. A distribution is right-skewed if its right tail is longer than its left and its mass leans to the left. We see that the PlanetLab load stands out as being more right-skewed than the others. All traces, however, show asymmetric right-skewed behavior, indicating that low demand is the most common state of the systems. Distribution models such as the Gaussian or Normal distribution assumes symmetry and will thus be inaccurate for all of the traces studied. The central moments are summarized in Table 1.

3.2 Dependence and Long Memory

One of the most common assumptions when studying time series and when sampling data to approximate distributions and densities is that the observations are IID. I.e. the sampled data points are independent and identically distributed. This allows the models to be trained once and then reused indefinitely when they have converged. It also simplifies the construction of confidence and prediction intervals. Because there is no bias in the samples they can be taken to be a good representation of the whole data set. The simplest way of testing dependence, seasonality and randomness of samples is to draw an auto-correlation function (ACF) plot with a series of increasing time lags. We study the correlations

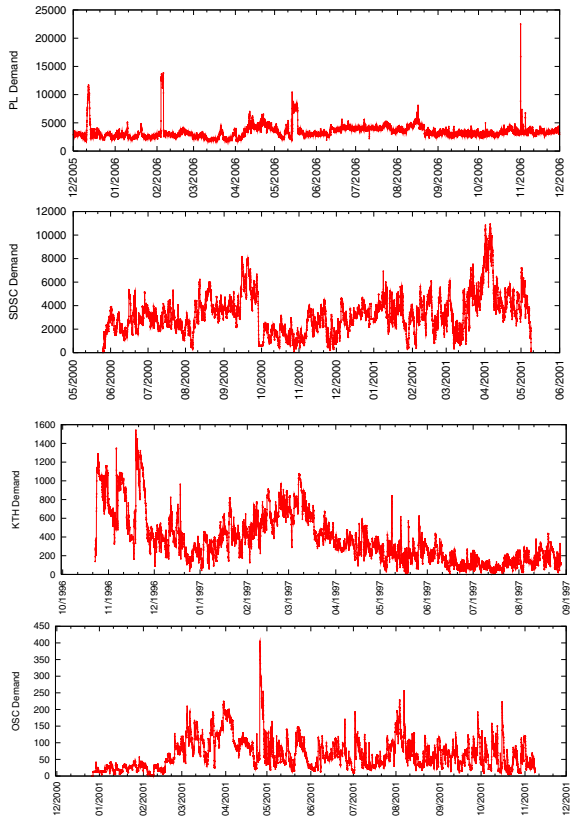

Figure 1: Demand History (Hourly)

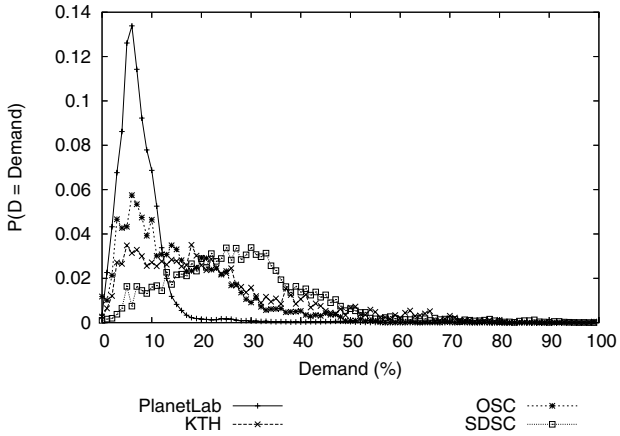

Figure 2: Demand Density

for lags up to 7 days. The plots in Figure 3, clearly show that the observations are not independent in time but rather show a strong and slowly decaying correlation to measures in the previous time interval. If the decay is slower than exponential the data is said to exhibit long memory. This is clearly the case for at least the KTH trace (within the weekly time lag). Long memory is a feature that gives standard auto-regressive models such as GARCH problems [18].

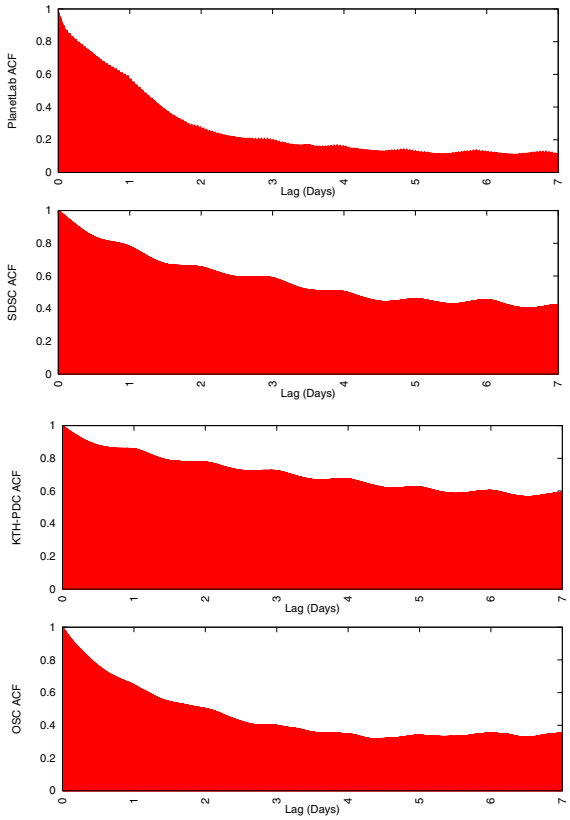

Figure 3: Auto-Correlation Functions

Another popular approach used to detect long-term correlations is the rescaled-range (R/S) analysis [19], which was influenced by Hurst's study of the Nile floods [14]. In general it provides a way to investigate how the values in a time series scale with time. The scaling index, known as the Hurst exponent is 0.5 if all the increments of the time series are independent. A series where a trend in the past makes the same trend in the future more likely is called presistent, and has a Hurst exponent greater than 0.5. Conversely, systems where past trends are likely to be reversed are called anti-persistent and scale with a Hurst exponent less than 0.5. If the R/S values (increment range, standard deviation ratio) for different time intervals are plotted against time on a log-log scale, the Hurst exponent appears as the slope of the fitted line. Figure 4 shows the R/S plot for the demand traces. A Hurst exponent around 0.92 fits all the traces under investigation, which indicates a very high correlation between past and future demand trends.

3.3 Heteroskedacity and Fat Tails

The last property that we investigate is the general volatility of the data which is crucial for making good risk assess-

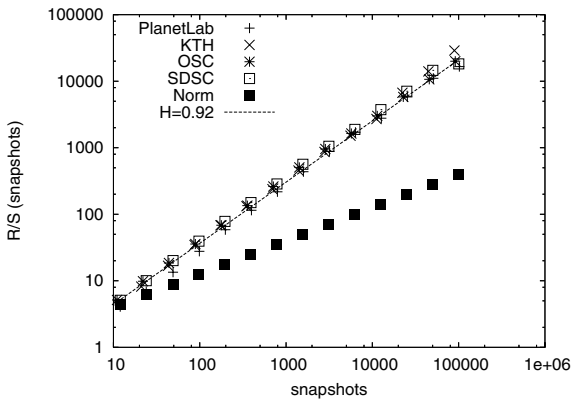

Figure 4: Rescaled-Range Dynamics and Hurst Exponent

ments. If the data is extremely skewed such as in power-law distributed data, both the mean and the variance can be infinite and thus some other measures of volatility need to be used. One popular approach is to look at the (absolute) log difference in the increments of the data. It turns out that even for very risky and volatile time series with power-law behavior like the stock-market, the absolute increments are predictable since high volatility data points tend to cluster [19]. A volatility graph showing the log-transformed increment differences over time is also another measurement of how Gaussian-like the data is. A Gaussian distribution produced by a Brownian-motion process has a near uniform distribution of increment sizes, whereas more unpredictable and thereby riskier processes have clusters of high volatility intermingled with lower volatility periods. In Figure 5 all of the traces show signs of changing volatility over time (heteroskedacity). High volatility instances also seem to be clustered. An analysis of how they are clustered is beyond the scope of this paper. The stock market has been shown to exhibit power-law scaling in the distribution of the volatility over time [19]. We therefore also look at the distribution tail behavior for our traces. A heavy-tailed or fat-tailed distribution will exhibit a longer tail of the complement of the cumulative distribution function (1-CDF) than the exponential distribution. According to this definition Figure 6 and Table 2 show that all traces are heavy-tailed in hourly volatility. PL and SDSC are also heavy-tailed in daily volatility.

This investigation of traces indicates that a multi-fractal time-scaling (trading time deformation) [18, 19] model may be appropriate. We, for example, note that the Hurst exponent obtained can be used to determine the fractal dimension [3], which is a typical measure of the *roughness* of the system in multifractal time-series. Figure 6 also indicates that a stretched exponential distribution [8] could be a good fit. However, an analysis of these more complicated distributional models are outside the scope of this paper.

4. PREDICTION APPROACH

In this section, we present our approach to providing accurate distribution predictions with an upper prediction bound. The method is agnostic to the model used to fit the time series data of demand, and the prediction error distribution.

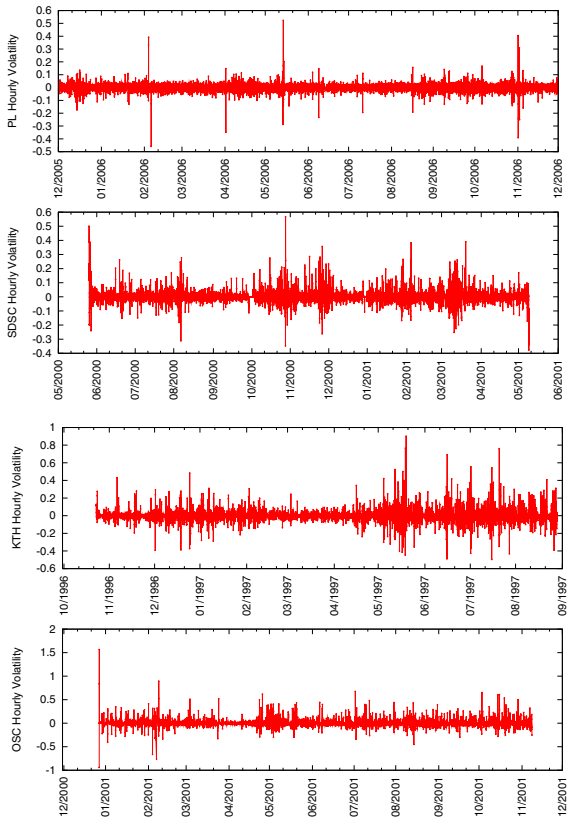

Figure 5: Hourly Demand Volatility

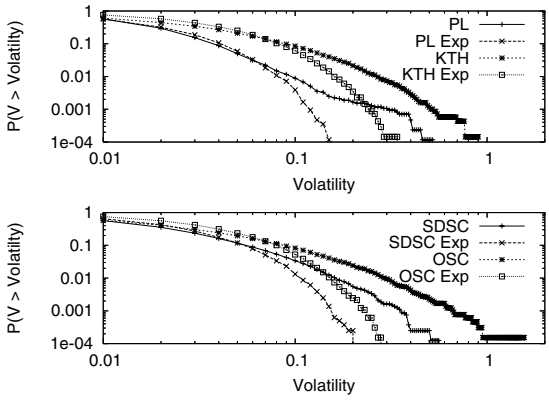

Figure 6: Heavy Tails for Hourly Demand Volatility

Table 2: Skewness of Volatility

	hour	day	week
PL	**7.45**	**4.55**	1.45
KTH	**4.40**	1.87	1.13
OSC	**6.41**	1.82	0.53
SDSC	**4.44**	**2.05**	1.63

There are two architectural components providing prediction capabilities: the *statistics collector* (SC), and the *prediction generator* (PG). The SC consumes a time series of prices and produces a statistical summary, which in turn is consumed by the PG. The statistical summary comprises instantaneous running (non-central) moments over configurable time horizons. In our case we provide hourly, daily and weekly summaries, as they correspond best to the length of the computational jobs run as well as the length of the horizon that is typically predictable. In addition to the moments, the summary also has the current price, a short history of moments for the most recent time periods, and the minimum and maximum measured price values.

The running non-central moments are calculated as follows:

$$\mu_{t,p} = \alpha\mu_{t-1,p} + (1-\alpha)x_t^p$$

where $\mu_{t,p}$ is the pth moment at time t, x_t the price at time t and $\alpha = 1 - \frac{1}{n}$, where n is the number of data points in the time horizon covered by the moments, and $\mu_{0,p} = x_0^p$. We refer to [22] for further details on how the central moments are calculated.

The PG component is instantiated with a predictor that uses the moments and the extremes to construct approximations of the cumulative distribution function (CDF), percent point function (PPF), and a function generating confidence intervals. The history of moments is used to construct prediction intervals.

Here we will describe a Gaussian (Norm), a Chebyshev (Cheb), and a sample-based predictor (Bench) which we use for benchmarking.

4.1 Gaussian Predictor

The Gaussian CDF (Φ) is readily available and can be calculated directly from the first two central moments. Since the inverse of the Gaussian CDF or PPF (Φ^{-1}) does not have a closed form, we use an approximation based on a rational minimax algorithm developed by Acklam [1]. The $100p$%-confidence interval is then calculated as $[\Phi^{-1}(0.5 - \frac{p}{2}), \Phi^{-1}(0.5 + \frac{p}{2})]$. An identical interval calculation can be done for all other predictors using their respective percent point functions. The prediction interval is calculated by applying Φ and Φ^{-1} on the history of moments.

4.2 Chebyshev Predictor

When predicting performance guarantees we are typically more interested in worst case scenario bounds as opposed to perfect data density fitting across all percentiles. One of the most prominent techniques to estimate bounds in probability theory is by means of the Chebyshev inequality [12], which states that:

$$P(|Y - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

where μ is the first central moment (mean) and σ the second central moment (standard deviation) of the underlying distribution. Rewriting the inequality as a CDF we get:

$$CDF(y) = 1 - \frac{1}{1+k^2}$$

where k is $\frac{y-\mu}{\sigma}$. For unimodal distributions we can tighten the bound further by applying the Vysochanskij-Petunin in-

equality [24], which for $k \geq \sqrt{\frac{8}{3}}$ gives:

$$CDF(y) = 1 - \frac{4}{9k^2}$$

Taking the inverse of the CDF we get:

$$PPF(p) = \begin{cases} \mu \pm \sigma\sqrt{\frac{1}{1-p} - 1} & , k < \sqrt{\frac{8}{3}}, \\ \mu \pm \sigma\sqrt{\frac{4}{9-p}} & , k \geq \sqrt{\frac{8}{3}}. \end{cases}$$

Since Chebyshev only gives us upper and lower bounds we cannot calculate the percentiles around the mean accurately, but this is not a great limitation in our case where we are primarily interested in worst-case scenario (upper) bounds.

The confidence and prediction intervals are calculated in the same way as in the Gaussian case.

4.3 Sample Bench Predictor

We use a benchmark predictor to compare our summary statistics predictors with a sample-based predictor. This predictor has access to the entire past history of data points and calculates the percentile points, cumulative distributions, and prediction bounds from the raw data sampled. The benchmark predictor could not be used in practice because of its prohibitive computational and storage requirements.

4.4 Multi-Host Predictions

We combine the results from Equation 1, Equation 2 and Equation 3 with our predictors to assess risk when making scheduling decisions across a set of hosts. For this purpose we extend an economic scheduling algorithm previously presented in [11, 22]. The purpose of this Best Response algorithm is to distribute a total budget across a set of hosts to yield the maximum utility for the user. The optimization problem, as seen by a user, is defined as:

$$\text{maximize } U = \sum_{j=1}^{n} w_j \frac{b_j}{b_j + y_j} \text{ subject to}$$

$$\sum_{j=1}^{n} b_j = bid, \text{ and } b_j \geq 0.$$

where w_j is the preference or weight for host j specified by the user, b_j is the bid that the user puts on host j, and *bid* the total budget of the user. We replace y_j, the spot price of host j, with the stochastic variable Y as modeled above. In Equation 1, which gives an expected performance value given a percentile and a (prediction) confidence level, we calculate Y with the PPF of the predictor. To calculate the *bid* given a performance level, a *guarantee* and a confidence level or to calculate the *guarantee* given a *bid* and a performance level, we numerically invert the Best Response algorithm. This allows us to probe different *bid* or *guarantee* levels until we get an acceptable performance match or we encounter budget constraints.

Users can use this model in a variety of ways. They can use the spot prices to determine whether they can meet their deadline, guarantee, and budget constraints, or if they should submit their job at a later time. Instead of the spot price, users can also use the statistical guarantees and prediction bounds to pick the hosts with the best sustained low-price behavior for long running tasks. We examine the effectiveness of the model for these use cases in the next section.

5. SIMULATION AND EXPERIMENT RESULTS

This section contains four different validators of our approach presented in the previous section. First, we run a simulation with generated random distributions against our predictors. Second, we run a prediction simulation using the compute cluster demand traces described in Section 3. Third, we run an experiment in our own live computational market cluster, Tycoon, comparing spot market scheduling with our extended risk mitigating scheduler described in the previous section. Finally, we run an efficiency experiment to measure the overhead incurred by the predictions.

5.1 Asymmetry Modeling Simulation

To validate the ability to approximate arbitrary skewed distributions we developed a generator capable of producing random data with distributions in a continuum from a right-skewed Pareto through Gaussian to a mirrored left-skewed Pareto with the first two moments kept stable and only the skewness varying. An example of distributions generated can be seen in the lower graph in Figure 7. The upper graph shows the result for the Cheb and Gauss predictors. As expected, Cheb gives better approximations for left and right-skewed distributions, whereas the Gauss predictor performs best for the near-symmetrical distributions (skewness near 0).

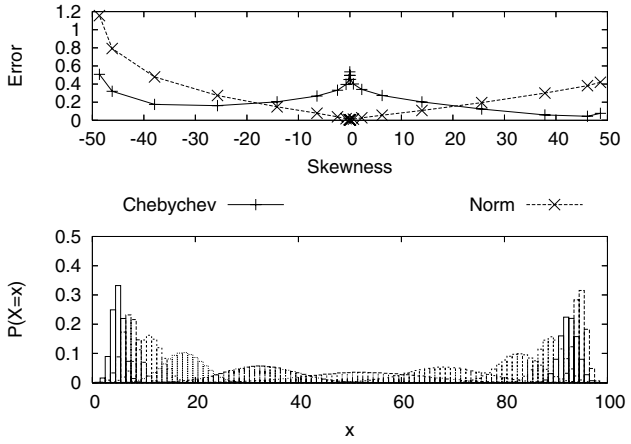

Figure 7: Fitting Skewed Distribution

5.2 Demand Prediction Simulation

We use the compute cluster demand traces from PlanetLab, KTH, SDSC, and OSC to study the ability of our prediction approach to give accurate risk assessments with the Cheb, Gauss, and Bench predictors. Recall that the Bench predictor simply uses all previous historical data to make PPF and prediction (confidence) interval estimates, whereas the other predictors base their estimates on the summary statistics generated by the statistics collector (SC) component (including running moments, and moment history).

The setup of the experiment is as follows. For each trace, we feed the time series data into the SC component configured for hourly, daily and weekly prediction horizons, and then try to make a prediction of the 95th percentile price

with a 90% confidence for the subsequent interval (for which no data is revealed) using the prediction generator (PG) instantiated with the Cheb, Gauss and Bench predictors. We then measure the delivered guarantee as the likelihood that at least 95% of the demand values are less than the predicted upper prediction confidence bound, which we denote the success rate (S). We also track the width of the prediction bound (B) as the difference between the actual 95th percentile demand and the predicted demand. The PG component is configured to track three historical running moment snapshots into the past of the first two non-central moments. The results are shown in Table 3. S and B are defined as:

$$S = P(\hat{f}(0.95) > f(0.95))$$

and

$$B = E\left(\frac{|\hat{f}(0.95) - f(0.95)|}{f(0.95)}\right)$$

where f is the actual percentile function of the price, and $\hat{f}$ is the predictor estimated percentile function of the price.

The Cheb predictor generates consistent and accurate success rates (S) for all traces across all prediction horizons. For daily and weekly horizons the prediction bound size (B) is in most cases very wide, which is likely to cause risk-averse users or users with a very limited budget to delay their job submissions. Both the Norm and the Bench predictors underestimate the risk grossly as seen by very low success rates, although the bounds are tight.

The Norm predictor would yield better success rates if we provided additional moment history snapshots. However, one of our requirements is to maintain system scalability and flexibility, so we must minimize the number of statistical summary points to reduce the size of the snapshots.

We note that using a horizon size of one and an infinite snapshot size in the SC component would make the summary statistics results converge to the results obtained by the Bench predictor.

5.3 Risk Mitigation Experiment

To experimentally validate our approach, we run a scheduling benchmark in a live Tycoon computational market. We submit jobs using the NorduGrid/ARC Grid meta-scheduler [23] and schedule locally using the extended Best Response algorithm described in Section 4. Tycoon uses the Xen virtual machine monitor [10] to host running jobs. Each job is run in a separate, dedicated, isolated machine.

The design rationale behind this experiment is to create a changing usage pattern that could potentially be predicted, and to study how well our approach adapts to this pattern under heavy load. We do not claim that the traffic pattern is representative of a real system. See the analysis in Section 5.2 to see how the Tycoon predictors handle real-world usage patterns.

The experiment consists of two independent runs with 720 virtual machines created on 60 physical machines during each 6 hour run. All jobs run on dedicated virtual machines that are configured based on the current demand, and job resource requirements. All jobs request 800MB of disk, 512MB of memory and $1 - 100\%$ CPU, depending on demand. The users are configured as shown in Figure 8. More specifically:

• **User 1 (Continuous)** continuously runs 60 parallel jobs with low funding on 30 physical hosts throughout the experiment run (6 hours). The set of hosts is static and separate from the bursty user's hosts.

• **User 2 (Bursty)** sporadically runs 60 highly funded 30 minute jobs every hour on the 30 physical hosts not used by User 1.

• **User 3 (Spot Market)** schedules and runs 30 jobs of 40 minutes each every hour based on spot market prices. The spot market user selects from any of the 60 hosts in the system.

• **User 4 (Predicting)** schedules and runs 30 jobs of 40 minutes each every hour based on the predicted 80th percentile prices using the PG/Cheb predictor consuming continuous statistical feeds from the SC component deployed on each compute node. The predicting user selects from any of the 60 hosts in the system.

All jobs run a CPU benchmark incrementing a counter with a frequency based on the allocated resource share. The value of the counter is our metrics for work completed. Both the spot market and predicting users have the same budget for purchasing resources.

The dynamic behavior of the system is as follows. The continous user's jobs run on the hosts in the left of Figure 8. The bursty user's jobs run on the right. The spot market user selects the host with the lowest spot price, so that the jobs will almost always run on one of the right hosts. On a few rare occasions, all of the right hosts will have a higher spot price than the left hosts. The predicting user selects based on historical prices. These tend to be lower for the left hosts since the bursty user avoids those hosts, so the predicting user's jobs will tend to use the left hosts.

The distribution of work completed by each job submitted by the spot market and predicting users are graphed in the top graph in Figure 9. The distribution for the predicting user is shifted to the right compared to the spot market

Table 3: Prediction Result of 95th Percentile with 90% Upper Prediction Bound. (S is the success rate and B the prediction bound.)

PL	Hour		Day		Week	
	S	B	S	B	S	B
Cheb	**0.93**	0.16	**0.95**	0.57	**0.93**	1.20
Norm	0.62	0.08	0.76	0.29	0.80	0.58
Bench	0.79	0.28	0.72	0.24	0.55	0.17
KTH	Hour		Day		Week	
	S	B	S	B	S	B
Cheb	**0.96**	0.38	**0.95**	0.92	**0.97**	0.91
Norm	0.87	0.22	0.87	0.47	0.76	0.44
Bench	**0.98**	3.25	**0.98**	1.97	**0.97**	1.40
OSC	Hour		Day		Week	
	S	B	S	B	S	B
Cheb	**0.94**	0.40	**0.94**	1.30	**0.94**	1.11
Norm	0.88	0.22	0.81	0.70	0.74	0.51
Bench	0.81	1.63	0.73	0.80	0.63	0.33
SDSC	Hour		Day		Week	
	S	B	S	B	S	B
Cheb	**0.95**	0.26	**0.96**	0.88	**0.95**	0.92
Norm	0.85	0.15	0.78	0.47	0.72	0.43
Bench	0.79	0.96	0.64	0.55	0.41	0.32

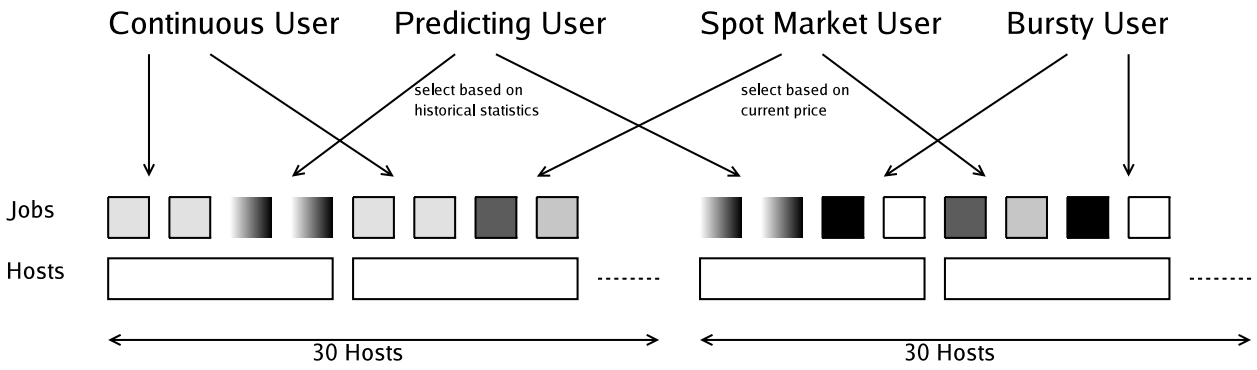

Figure 8: Risk mitigation experimental setup

user, showing that the predicting user finishes more work. The bottom graph shows the cumulative distribution function (CDF) of the work completed by the two users. The area between the two plots shows that fewer predicting jobs were impacted by the bursty user. On average, the jobs of the predicting user performed 22.7% more work. The spot market user's jobs on the far right of the CDF were able (by chance) to run on hosts at times when none of the bursty user's jobs were running. This is why they were able to complete so much work. In contrast, the predicting user almost never selects such hosts because it predicts (accurately in most cases) that the bursty user will soon return. Thus, risk mitigation both increases average performance and reduces variability in performance, even under heavy and spiky load, given that the spiky behavior is predictable over some time horizon (1 hour in this case).

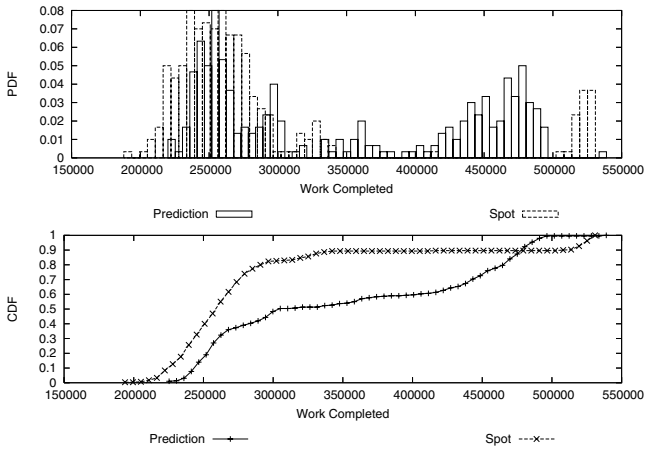

Figure 9: Risk Mitigation using Percentile Predictions

5.4 Prediction Efficiency Experiment

As a final experiment, we evaluate the overhead imposed by the prediction implementation presented in this paper compared to the standard spot market budgeting algorithm used in the previous experiment. The standard algorithm involves the following two steps: (1) get live host spot-market price information, (2) evaluate the bid distribution across the hosts given a total budget to maximize the aggregate performance. The prediction implementation extends step 1 by retrieving the summary statistics required to calculate the prediction bounds, and then extends step 2 by calculating the optimal bid distribution given a desired guarantee level and a given prediction confidence bound level. In the experiment the algorithms were run against a cluster of 70 hosts, using a guarantee level of 95% and a confidence bound of 90%. We interleave 400 runs using the two algorithms and measure the round-trip time of each operation.

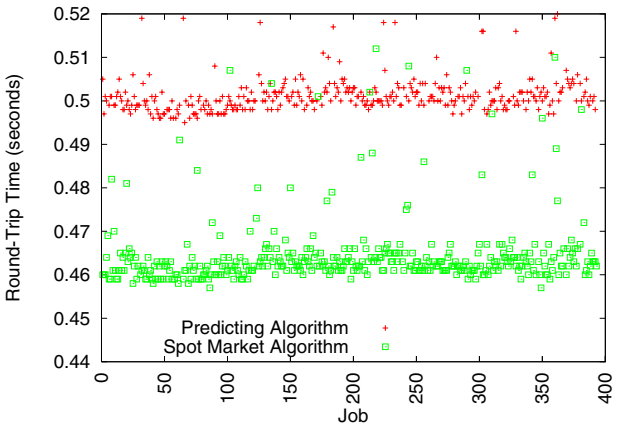

Figure 10: Round-Trip Times of Spot-Market vs Prediction-Based Host Selection and Budgeting

Figure 10 shows the results. The mean round-trip time for spot-market budgeting is .46 seconds and the mean for prediction is .5 seconds. The impact of this overhead on actual running time depends on how frequently the budgeting process is run. In the experiment in Section 5.3, we budget once an hour, so the overhead of the prediction algorithm over the spot market one is $(.5 - .46)/3600 = .001\%$. In other words, the overhead for 70 hosts is negligible, indicating that the algorithm will scale to tens of thousands of hosts.

6. RELATED WORK

MacKie-Mason et. al. [17] investigate how price predictors can improve users' bidding strategies in a market-based re-

source scheduling scenario. They conclude that simple predictors, such as taking the average of the previous round of auctions, improve expected bidder performance. Although the goal of this work is similar to ours, they investigate a different combinatorial allocation scenario where first price winner-takes-it-all auctions are employed, as opposed to the proportional share allocation in our work. Nevertheless, their results are encouraging.

Another use of economic predictions is described by Wellman et. al. [26], where bidding agents use the expected market clearing price in a competitive or Walrasian equilibrium. They employ *tatonnement* which involves determining users' inclination to bid a certain value given a price-level. Wellman et. al. compare their competitive analysis predictor to simple historical averaging and machine learning models. They conclude that strategies that consider both historical data and instance-specific data provide a competitive advantage. The conditional probability of price dynamics given a price-level would be additional useful information in our model. However, this is probably impractical in large-scale systems with users entering and leaving the market at will, and with large real-valued price ranges, so we assume this behavior is incorporated in the price history itself.

Wolski et. al. [28] describe the Network Weather Service (NWS) which has been used in large-scale Grid deployments to monitor compute jobs. Our work differs from NWS in both how statistics are collected and stored and how predictions are computed. NWS uses a multi-service infrastructure to track, store and distribute entire time-series feeds from providers to consumers via sensors and memory components (feed history databases). Our solution only maintains summary statistics and therefore is more light-weight. No persistent storage or searching infrastructure is required. For prediction, NWS uses simple moving average with static parameters. We use more general predictors that can handle any dynamics and adapt their parameters automatically. In addition, the focus in [28] is on predicting queue wait times, whereas we focus on predicting actual demand.

Brevik et. al. [4] present a Binomial Method Batch Predictor (BMBP) complementing NWS [28]. The approach is to assume that each observed value in the time-series can be seen as an independent draw from a Bernoulli trial. The problem is that this does not account for time correlations, which we have found to be substantial in our analysis. Brevik addresses the correlation by first detecting structural changes in the feed when BMBP generates a sequence of bad predictions and thereafter truncating the history which the predictor model is fit against. Our approach is to leverage the correlation by using biased samples of the most recent time intervals, which result in dynamic adaptation of 'structural' changes in the feed. The problem of monitoring and fixing prediction problems a posteriori as in BMBP is that the detection mechanism is somewhat arbitrary and a structural failure of the model could result in great losses, which could defeat the purpose of providing risk mitigating predictions [18].

Our prediction interval calculation was inspired by Haan and Meeker [13] but they also assume that random independent samples are drawn and that a large number of sample data points are used to yield tight prediction bounds. Neither of these two assumptions are true in our scenario. Our calculation of the prediction interval can be seen as more in the spirit of the simple empirical intervals proposed by Williams and Goodman [27]. Their empirical source is the previous sample point, whereas, we use summary statistics as input to the empirical predictions. This allows us to cover larger prediction horizons with greater confidence using fewer data points.

The data analysis of the distribution characteristics in Section 3 was inspired by the work by Mandelbrot on modeling risk in volatile markets [19]. The fat-tail behavior of the hourly volatility (not daily or weekly across all the traces) fits well with the volatility Mandelbrot has seen in the cotton-price, Deutschmark-Dollar exchange rate, and the stock price market dynamics, which he calls 'wild' randomness or chance.

7. CONCLUSIONS

All of the demand traces studied show non-Gaussian properties, which called for more generic distribution-free predictors. The clear correlation between subsequent hourly, daily and weekly time intervals of the traces suggests that the typical IID assumption is not valid and would lead to risk underestimation. This leads us to a model based on dynamically tracking running moments and the most recent snapshots of these moments instantiated by a worst-case bound, Chebyshev inequality influenced distribution estimator. Although this predictor does not generate tight prediction bounds for daily and weekly predictions, it is consistent in the confidence levels promised across all traces investigated, which makes it a good general indicator of the model uncertainty and reliability of the predictions. In highly volatile environments, making point predictions into the future is not possible with any level of accuracy, but high volatility periods are typically clustered. Thus, accurately estimating model uncertainty helps users to decide 1) whether to delay running their jobs until after the 'stormy' period has passed, or if they do decide to run, 2) how much insurance funding to spend.

The Bench predictor shows how poor predictions can be if one only relies on the available past history and ignores time correlations. Similarly, the Norm predictor exemplifies how inaccurate predictions can be if symmetric Gaussian distributions are wrongly assumed. The distribution agnostic Cheb predictor, on the other hand, provides superior predictions in cases where demand was heavily skewed and the volatility spiky, e.g. in the PlanetLab trace.

We have seen that our prediction approach can easily be deployed in scheduling scenarios where the most reliable hosts, delivering a good sustained performance over time, need to be picked for long-running jobs.

This work has focused primarily on helping consumers spend the right amount of money when purchasing resource shares, but the prediction approach with SC/PG/Cheb is general enough to be used by brokers pricing options or reservations as well, which is the focus of future work.

We would also like to study the effects different mixes of prediction, spot-market and reservation usage patterns have on the overall system efficiency and fairness.

8. ACKNOWLEDGMENTS

We thank our colleagues Scott Clearwater, Bernardo Huberman, Li Zhang, Fang Wu, and Ali Ghodsi for enlightening discussions. This work would not have been possible without the funding from the HP/Intel Joint Innovation

Program (JIP), our JIP liason, Rick McGeer, and our collaborators at Intel, Rob Knauerhase and Jeff Sedayao. We are grateful to Vivek Pai at Princeton University for making the PL trace available and helping us interpret it; Travis Earheart and Nancy Wilkins-Diehr at SDSC for making the SDSC trace available; and Lars Malinowsky at PDC for providing the KTH trace.

9. REFERENCES

[1] An Algorithm for Computing the Inverse Normal Cumulative Distribution Function. http://home.online.no/~ pjacklam/notes/invnorm/, 2007.

[2] Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/, 2007.

[3] M. Bodruzzaman, J. Cadzow, R. Shiavi, A. Kilroy, B. Dawant, and M. Wilkes. Hurst's rescaled-range (r/s) analysis and fractal dimension of electromyographic (emg) signal. In *Proceedings of IEEE Souteastcon '91*, pages 1121–1123, Williamsburg, VA, USA, 1991. IEEE.

[4] J. Brevik, D. Nurmi, and R. Wolski. Predicting bounds on queuing delay for batch-scheduled parallel machines. In *PPoPP '06: Proceedings of the 2006 ACM Principles and Practices of Parallel Programming*, New York, NY, USA, 2006. ACM.

[5] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Proceedings of the IEEE, Special Issue on Grid Computing*, 93(3):479–484, March 2005.

[6] Chaki Ng and Philip Buonadonna and Brent N. Chun and Alex C. Snoeren and Amin Vahdat. Addressing Strategic Behavior in a Deployed Microeconomic Resource Allocator. In *Proceedings of the 3rd Workshop on Economics of Peer-to-Peer Systems*, 2005.

[7] S. Clearwater and B. A. Huberman. Swing Options. In *Proceedings of 11th International Conference on Computing in Economics*, June 2005.

[8] S. Clearwater and S. D. Kleban. Heavy-tailed distributions in supercomputer jobs. Technical Report SAND2002-2378C, Sandia National Labs, 2002.

[9] David C. Parkes and Ruggiero Cavallo and Nick Elprin and Adam Juda and Sebastien Lahaie and Benjamin Lubin and Loizos Michael and Jeffrey Shneidman and Hassan Sultan. ICE: An Iterative Combinatorial Exchange. In *Proceedings of the ACM Conference on Electronic Commerce*, 2005.

[10] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer. Xen and the Art of Virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2003.

[11] M. Feldman, K. Lai, and L. Zhang. A Price-Anticipating Resource Allocation Mechanism for Distributed Shared Clusters. In *Proceedings of the ACM Conference on Electronic Commerce*, 2005.

[12] W. Feller. *An Introduction to Probability Theory and its Applications, volume II*. Wiley Eastern Limited, 1988.

[13] G. J. Hahn and W. Q. Meeker. *Statistical Intervals: A Guide for Practitioners*. John Wiley & Sons, Inc, New York, NY, USA, 1991.

[14] H. Hurst. Long term storage capacity of reservoirs. *Proc. American Society of Civil Engineers*, 76(11), 1950.

[15] L. V. Kale, S. Kumar, M. Potnuru, J. DeSouza, and S. Bandhakavi. Faucets: Efficient resource allocation on the computational grid. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)*, pages 396–405, Washington, DC, USA, 2004. IEEE Computer Society.

[16] K. Lai. Markets are Dead, Long Live Markets. *SIGecom Exchanges*, 5(4):1–10, July 2005.

[17] J. K. MacKie-Mason, A. Osepayshvili, D. M. Reeves, and M. P. Wellman. Price prediction strategies for market-based scheduling. In *ICAPS*, pages 244–252, 2004.

[18] B. Mandelbrot, A. Fisher, and L. Calvet. The multifractal model of asset returns. In *Cowles Foundation Discussion Papers: 1164*. Yale University, 1997.

[19] B. Mandelbrot and R. L. Hudson. *The (Mis)behavior of Markets: A Fractal View of Risk, Ruin, and Reward*. Basic Books, New York, NY, USA, 2004.

[20] L. Peterson, T. Anderson, D. Culler, , and T. Roscoe. Blueprint for Introducing Disruptive Technology into the Internet. In *First Workshop on Hot Topics in Networking*, 2002.

[21] O. Regev and N. Nisan. The Popcorn Market: Online Markets for Computational Resources. In *Proceedings of 1st International Conference on Information and Computation Economies*, pages 148–157, 1998.

[22] T. Sandholm, K. Lai, J. Andrade, and J. Odeberg. Market-based resource allocation using price prediction in a high performance computing grid for scientific applications. In *HPDC '06: Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing*, pages 132–143, June 2006. http://hpdc.lri.fr/index.php.

[23] O. Smirnova, P. Erola, T. Ekelöf, M. Ellert, J. Hansen, A. Konsantinov, B. Konya, J. Nielsen, F. Ould-Saada, and A. Wäänänen. The NorduGrid Architecture and Middleware for Scientific Applications. *Lecture Notes in Computer Science*, 267:264–273, 2003.

[24] D. F. Vysochanskij and Y. I. Petunin. Justification of the 3 sigma rule for unimodal distributions. *Theory of Probability and Mathematical Statistics*, 21:25–36, 1980.

[25] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A Distributed Computational Economy. *Software Engineering*, 18(2):103–117, 1992.

[26] M. P. Wellman, D. M. Reeves, K. M. Lochner, and Y. Vorobeychik. Price prediction in a trading agent competition. *J. Artif. Intell. Res. (JAIR)*, 21:19–36, 2004.

[27] W. Williams and M. Goodman. A simple method for the construction of empirical confidence limits for economic forecasts. *Journal of the American Statistical Association*, 66(336):752–754, 1971.

[28] R. Wolski, G. Obertelli, M. Allen, D. Nurmi, and J. Brevik. Predicting Grid Resource Performance On-Line. In *Handbook of Innovative Computing: Models, Enabling Technologies, and Applications*. Springer Verlag, 2005.

[29] R. Wolski, J. S. Plank, T. Bryan, and J. Brevik. G-commerce: Market formulations controlling resource allocation on the computational grid. In *IPDPS '01: Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01)*, page 10046.2, Washington, DC, USA, 2001. IEEE Computer Society.

[30] F. Wu, L. Zhang, and B. A. Huberman. Truth-telling Reservations. http://arxiv.org/abs/cs/0508028, 2005.

[31] L. Xiao, Y. Zhu, L. M. Ni, and Z. Xu. Gridis: An incentive-based grid scheduling. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, page 65.2, Washington, DC, USA, 2005. IEEE Computer Society.