Admission Control in a Computational Market

Thomas Sandholm
School of Information and Communication Technology
KTH – Royal Institute of Technology
SE-16440 Kista, Sweden
sandholm@kth.se

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

Scott Clearwater
P.O. Box 1252
Los Altos, CA 94023, USA
clearway@comcast.net

Abstract

We propose, implement and evaluate three admission models for computational Grids. The models take the expected demand into account and offer a specific performance guarantee. The main issue addressed is how users and providers should make the tradeoff between a best effort (low guarantee) spot market and an admission controlled (high guarantee) reservation market. Using a realistically modeled high performance computing workload and utility models of user preferences, we run experiments highlighting the conditions under which different markets and admission models are efficient. The experimental results show that providers can make large efficiency gains if the admission model is chosen dynamically based on the current load, likewise we show that users have an opportunity to optimize their job performance by carefully picking the right market based on the state of the system, and the characteristics of the application to be run. Finally, we provide simple functional expressions that can guide both users and providers when making decisions about guarantee levels to request or offer.

1. Introduction

In large-scale shared systems, such as cross-organizational Grids, the *best-effort* provisioning model has dominated because of its scalability, high utilization, and high availability. *Elastic* applications that can tolerate variability in performance thrive in this model. Examples include batch-processing applications like data mining, and data warehousing. Other, *inelastic* applications require more stringent *quality-of-service* (QoS) assurances for their users. Examples of these are interactive applications like web applications, and media servers. These applications require an admission control mechanism to prevent high load from violating QoS assurances. However, admission control reduces scalability, reliability, and utilization and introduces the risk of being rejected admittance.

The key difference between best-effort and admission control is that best-effort imposes more costs (in the elasticity requirement) on applications whereas admission control imposes more costs (in scalability, reliability, utilization, and rejections) on the system provider. Nonetheless, for many applications, the cost to implement elasticity is high. As a result, maximizing system efficiency requires both models, but also incentives for applications to use the best-effort model. Economic systems (e.g., [12], [27], [23], [8], [18]) use payment as the incentive. This encourages applications that can implement elasticity to use best-effort while also accommodating those that cannot. In addition, admission control in a priced system provides an opportunity for the provider to do price discrimination and thereby sustain its profitability.

We examine such a hybrid system (based on the Tycoon[18] market-based resource allocator) in this paper. Having two resource models in a system introduces several questions:

What service model should an application use? Most applications are neither purely elastic nor purely inelastic. Instead, a typical application has a specific tolerance for delay. How easily the system can meet delay requirements varies over time as the system load changes. Moreover, an application must balance its delay tolerance with its willingness to pay for performance and the current cost of resources, which constantly changes in a market-based system.

How much of a resource should a service provider allocate to each model? The goal of the service provider is to maximize its profits. It does this by packaging its limited resources in a ratio of best-effort and admission control at prices that are desirable to applications. The ratio and the prices change over time as applications come and go. The provider must also increase the price of admission control resources to include the expected opportunity cost of future rejected admissions.

Previous studies of admission control have been in non-economic systems, been tied to a specific scheduling discipline, and focused on simulations driven by simplified distributional models. Instead, we focus on the admission control problem in an economic system and our work is independent of scheduling. Moreover, we construct and use workload models derived from a HPC production cluster and we evaluate an admission control broker in a live computational market cluster, using all the components of the production system.

Our key contribution is a set of admission control models based on statistical properties of the demand. Admission decisions rely both on expected risk of future rejections, and existing commitments. The risk estimator uses recent price history to calculate a statistical bound (Chebychev upper bound) on the likelihood of a certain deviation from the average demand. The price for a reservation is thus fully dynamic and correlated with the demand, while inelastic applications still obtain high guarantees.

In our experiments, we measure *economic efficiency*, which is the ratio of the utility obtained to the optimal utility (with a theoretical admission model that has full knowledge of future jobs). We show that the economic efficiency of a system that only provides best-effort is 80 per cent at low load, but only 55 per cent at high load. In the high load scenario, inelastic applications have only 35 per cent efficiency, while elastic applications have 90 per cent efficiency. In contrast, an admission control model offering absolute guarantees is 75 per cent efficient, regardless of load and application elasticity. Hence, providers have an incentive to choose and partition the admission model based on resource contention, and users have an incentive to pick an admission model that fits their job preferences. The results presented in this paper aid both the provider and the user in making these decisions dynamically.

The rest of the paper is structured as follows. In Section 2 we describe the workload model used in the experiments. The admission models are presented in Section 3. Section 4 discusses the experiments, Section 5 analyzes the experiment results, Section 6 compares our approach to related research, and finally Section 7 provides concluding remarks.

2. Workload Model

The goal of the workload model is twofold. It should be representative of workloads observed in a shared cluster production system, and it should enable efficient study of parameter spaces in our system. All jobs are allocated and consume real system resources in a full deployment of a computational market. Therefore, the workloads must be short enough to make the results easily reproducible while capturing as much of the statistical traits of the original trace as possible. Pure simulations [1] are easier to evaluate statistically, but our main contribution in this work is the evaluation of a real, distributed deployment.

Instead of using over-simplified assumptions and generalizing behavior across different clusters and sites, we chose to capture individual traces in more precise models. Here we present the results from using the SDSC (San Diego Supercomputer Center) Blue Horizon cluster trace from April 2000 to January 2003, containing 223402 jobs [11].

Distributional models for job inter-arrival time (IAT), runtime (RUN), and number of CPUs used per job (CPU) were constructed directly from the trace. The techniques used to model distributions are very generic and can be applied to a wide range of workload traces. We chose to focus solely on the SDSC trace for three reasons. First, the trace is the longest one available from the parallel workloads archive, and thus gives us the best statistical stability. Second, our main interest is in studying the parameter space of different admission policies under varying load, and hence utilizing a single workload model simplifies the matrix of configurations to be investigated. Third, the trace showcases characteristics that have been observed in many other HPC traces [11].

To synthesize workloads with the same statistical properties as the original trace, we construct a functional representation of the inverse of the cumulative distribution function (CDF), sometimes called the percent point function (PPF) or the quantile function. Many distributions, such as the Normal or Gaussian distribution do not have simple analytical representations of the PPF, which also effects our choice of model. To generate the desired workload the PPF is applied to a series of uniformly distributed random variables in the interval $(0, 1)$.

Inter-Arrival Time. The inter-arrival times were found to follow two different regimes. This bimodality was also found by just studying the last $1/10$ of the trace as well as in parallel archive workload traces from KTH and OSC, not presented here. The pattern found was that jobs submitted within about 10 seconds of each other were overrepresented and followed a separate distribution compared to all other jobs. Standard distributional models do not handle multimodality and we therefore had to represent this distribution

[1] see [25] for a similar simulation-based evaluation

with a mixture model. Each distribution in the mixture was fit to the data using a standard maximum-likelihood estimation (MLE) algorithm. The resulting model, which we call hyper log-weibull (HypWeib), is represented as follows:

$$Z = \alpha X + (1 - \alpha)e^Y \quad (1)$$

where Z is the random variable of the IAT distribution, and α was found to be 0.29. Furthermore, if $X \sim Weibull(16.1489, 1.3462)$ and $Y \sim Weibull(5.93123, 4.95969)$ then $Z \sim HypWeib$. The Weibull CDF is easy to invert arithmetically into a PPF, and therefore our HypWeib distribution is also easily representable as a PPF, which can be used to generate a synthetic IAT workload.

Runtime. The runtime data was also found to be bimodal. Jobs with runtimes less than 30 seconds followed a different distribution than the rest. Since these jobs are unlikely to have produced any useful work and constitute less than 7 per cent of the data, we do not represent them in our runtime model. No standard distribution was found with the maximum-likelihood algorithm that approximated the empirical CDF satisfactorily, therefore we used a polynomial linear least-squares fit of the CDF. To simplify the arithmetic inversion into a PPF we used a 2^{nd} order polynomial as follows:

$$CDF(x) = -0.0125x^2 + 0.3018x - 0.8184. \quad (2)$$

We call this fit the PolyLin distribution in the discussion below.

Job Size. From the density function of number of CPUs per job (CPU) it was apparent that values with base 2 were overrepresented. Only about 1 per cent of the jobs did not follow this pattern. We therefore neglect these jobs in our model for simplicity. Again the log base 2 transformed values do not follow a standard CDF well, but since it is a discrete variable and the number of different values with more than 3 per cent density are only six, the distribution which we call Binary Bin (BinBin) can be easily and accurately represented as a histogram as follows:

$$Z = 2^{X+2} \quad (3)$$

where Z is the random variable in the CPU distribution, X is distributed as the histogram density function $p(x_i) = v_i$, where x is the vector $\{1..6\}$ and v is the vector $\{0.508, 0.144, 0.126, 0.137, 0.048, 0.035\}$. Alternatively, this decline in binary exponents can be modeled as a Zipf(1.4), with a slightly worse fit but with one instead of six parameters.

Model Evaluation. To obtain quantitative statistics for how well the models fit the data we utilize the Two-sided Kolmogorov-Smirnov (KS) test. A bootstrap technique [9] is used to mimic the experiment setup of small representative sample runs. We draw 100 random samples from the empirical and our synthesized workloads, before comparison. For each pair of samples we calculate the KS value (max absolute difference between CDFs), and test the null hypothesis that the two samples are drawn from the same distribution at a 5 per cent significance level. We then record the success-rate, where the null hypothesis could not be rejected, and the average KS values for the sample tests as well as a KS test performed on the entire data sets. As a benchmark we also perform the hypothesis test on the trace with itself. It is commonly assumed that the job submissions follow a Poisson process (submissions are spread uniformly and independently over some time interval), which results in the IAT being exponentially distributed. As an additional comparison we therefore also model the IAT with an MLE fitted Exponential (Exp) distribution with mean 378.648. Similarly power-law distributions have been observed for process runtimes so we also fit the runtime data to a Pareto (Par) distribution with location parameter 1.77332, scale parameter 213.171 and threshold parameter 0, again obtained using MLE. Table 1 summarizes the goodness-of-fit results. From these results we can conclude that our three models, HypWeib, PolyLin, and BinBin, all provide accurate fits to the SDSC trace data. The KS test clearly rejects the Exp model as a good representation of the IAT distribution. Furthermore, we calculate the coefficient of variance ($CV = \sigma/\mu$) of the IAT data to be 3.86, which also rules out a Poisson process, which has an expected CV of 1. The Par model is a good fit to the runtime data, but not as good as the PolyLin fit. PolyLin provides a slightly worse fit than HypWeib and BinBin most likely due to the fact that we ignored 7 per cent of the shortest running jobs for simplicity reasons. To summarize, when taking 1000 samples (with replacement) of 100 values each from the synthesized distribution we can get representative values, according to the KS statistic at a 5 per cent significance level, in 87 per cent of the samples or more for all of our models.

Job Value. No information about user-specified job valuations are available in the trace that we study. Therefore, a set of standard distribution models are used. Three distributions, equal importance (Equal), normal (Norm), pareto power-law (Pareto) are modeled. Equal importance distributions occur when all jobs and all users have the same importance. This model is exemplified in the PlanetLab network. A normal distribution is the most common assumption as it can represent all populations produced by aggregating individually independent random variables with finite mean and variance, in accordance with the central limit theorem. Finally, the Pareto distribution was originally used to model the distribution of incomes, and similar power-law relationships have been observed in various large-scale network metrics, such as popularity of web sites.

Correlations. Supercomputer jobs are known to exhibit long-term correlations (or long memory) over time that

Table 1. Goodness-of-fit Results using Kolmogorov-Smirnov tests against the Trace data.

KS Data 1	KS Data 2	KS success-rate	KS complete data	KS sample mean
Trace IAT	Trace IAT	0.948	0	0.1161
HypWeib	Trace IAT	0.897	0.0557	0.1337
Exp	Trace IAT	0.034	0.2406	0.2901
Trace RUN	Trace RUN	0.944	0	0.1174
PolyLin	Trace RUN	0.869	0.0718	0.1371
Par	Trace RUN	0.818	0.1075	0.1565
Trace CPU	Trace CPU	0.989	0	0.0861
BinBin	Trace CPU	0.982	0.0165	0.0880
Zipf	Trace CPU	0.974	0.0644	0.1040

could lead to periods of anomalously high load [16]. A metric often used to quantify the correlation is the Hurst [13] and Mandelbrot [20] R/S statistic or Hurst exponent. A memoryless process (such as white noise) would have a Hurst exponent of 0.5 whereas processes exhibiting long memory would have exponents close to 1. Exponents less than 0.5 indicate *anti-correlation*, i.e. a past trend is likely to be reversed. We measured a Hurst exponent of 0.735 for the IAT series in the SDSC data, indicating moderate long term correlations, which corresponds well to previous work [16]. To adequately represent a given Hurst exponent in synthesized data, a very large number of data points need to be generated, which makes it impractical for our experiments. Instead we run our experiments under a couple of different load and IAT configurations to represent both regular operation periods and high load periods. We also ran some experiments where we induced short-term time correlations in the IAT value series with a simple sort and permute algorithm, and found that the results were largely the same, with the exception that a purely statistical admission model performed slightly better, as expected. We further note that no significant cross-correlations were found between the inter-arrival time, runtime and job size properties. Due to space limitations and the complexity of a thorough treatment, the correlation issue is left out-of scope and as the focus of future work.

3. Admission Models

In this section three different admission models are described, *best effort* (BE), *statistical admission control* (SAC), and *capacity admission control* (CAC). All of these models take a contract request and produce either an accepted or a rejected contract. The contract request has two parts, *service level requirements* and *resource requirements*. The service level requirements contain the budget a consumer is willing to pay for the resources, as well as total number of work units (CPU cycles in our experiment) needed per node, parallelism (number of nodes), deadline and type of contract (BE, SAC or CAC). The resource requirements specify detailed min and max bound preferences for resources such as CPU, memory, disk, and bandwidth. The resource requirements are enforced and continuously evaluated by the market-based resource allocator (Tycoon's Xen virtualization layer), and the service level requirements are enforced at submission time and evaluated at completion time by the admission model implementation. An approved contract contains a list of nodes selected to run the job and their individual funding levels.

Best Effort. In the best effort model the contract is only rejected if the current spot market price is too high to get the resources specified in the resource requirements part of the contract. The service level part is only used to validate the contract *a posteriori*. Existing jobs can be preempted by new arrivals that pay more. Note that preemption here refers to performance or resource share degradation only, not necessarily that the preempted job is stopped. The resource share obtained is:

$$q = \frac{b}{b+c} \tag{4}$$

where b is the spending rate (e.g. \$/s) derived from the budget and the deadline, and c is the current cost or price of the resource defined as: $c = \sum_i b_i$ where b_i is the current spending rate of consumer i.

Statistical Admission Control. In the statistical admission control model the contract is rejected if the budget is less than an estimated future percentile of the price, or if the current spot market price is too high to get the required resources. Existing jobs can be preempted. The resource share obtained with this model will not drop below q_{min} with a statistical guarantee g where

$$q_{min} = \frac{b}{b + F^{-1}(g)} \tag{5}$$

and $F^{-1}(g)$ is an estimate of the inverse of the cumulative price distribution function for a guarantee in the interval $(0..1)$. As an approximation of F^{-1} we use the Chebychev bound for a given mean and standard deviation of the price. More details on this technique can be found in [24].

Capacity Admission Control. The capacity admission control model performs the same statistical check as in the SAC model to ensure that a minimum price which is higher than the spot market price is paid for a capacity controlled contract. If the statistical check succeeds, an additional check is made to ensure that no currently active contracts are violated. A contract violation is detected by checking the resource shares obtained for all running jobs if the new job were to be admitted. If the share is below what is required to process the total number of work units for any job, the contract of that job is violated. Existing jobs can thus not be preempted in this model. The admission test (which must be true for a contract request to be accepted) is:

$$\forall s \in S : \sum_{h}^{n} \frac{b_h(s)}{b_h(s) + b_h(r) + c} \geq q_s \qquad (6)$$

where S is the set of all existing contracts including the requested contract, n is the number of hosts, $b_h(s)$ is the bid on host h in contract s, $b_h(r)$ is the bid on host h in the requested contract r, and q_s is the minimum performance share promised in contract s. Enforcement may be done on a host by host basis or across all hosts in the contract. We chose the latter in our experiments to allow hosts with a higher than promised performance to compensate for slower hosts in the same contract, and thereby cause fewer rejections.

4. Experiments

4.1. Configuration and Setup

Each job runs a CPU intensive benchmark application until the runtime has expired or the job has completed. Each user gets a budget of $100 to split among its jobs, according to relative value, runtime, and size (CPUs). After the job has finished we record the number of work units completed, which is directly proportional to total number of CPU cycles consumed. The SAC and CAC admission controllers both use the 60 per cent Chebychev bound as price rejection threshold (users who are not willing to pay more are rejected).

A work plan generated using the workload model described in Section 2 serves as input to each experiment run. The work plan contains parameters for each job to be run including: job id, IAT, runtime, CPUs, value, model, and user id. The model parameter is best effort (BE), statistical admission control (SAC), or capacity admission control (CAC).

An experiment run comprises a 2 hour trace of 50 jobs executed on a market-based cluster of 15 hosts and 30 CPUs. 10 users submit 5 jobs each with 4-10 CPUs/job, 2-16min runtime/job, and 30s-8min IAT/job. An identical trace is run with the BE, SAC, and CAC admission models. In order to make statistical claims about differences among the admission policies, each experiment run is repeated with five random traces generated with the same workload model configuration. The rationale behind this setup is to capture both aleatory (randomness within a distributional model) and epistemic (variations due to external factors, in our case live system behavior not captured in our model) uncertainty common in risk modeling [21].

The results from five different workload model configurations are presented below: Pareto job value distribution under low load, Pareto job value distribution under high load, equal job values under high load, Normal job value distribution under high load, and finally a random admission benchmark configuration under high load. A single run of all the experiments including epistemic repetitions, thus took about 150 hours (6 days and 6 hours), which explains why we limited the repetitions to five.

The different load levels were obtained by generating traces with minimum/mean IAT $90/153.66$, and $30/92.36$ seconds for low, and high load respectively. The load levels were set to result in moderate and extensive contention for resources. As the moderate contention workload is less sensitive to the job characteristics in general and the job valuation in particular, we only present the results from the Pareto distribution under low load here.

4.2. Metrics

We are mainly interested in the economic efficiency and fairness of the system, but some basic properties of the admission controller must be met. In particular an admission controller which rejects too many requests will cause underutilization of resources. Similarly, an admission controller that does not decrease the number of contract violations is of little value. We therefore compare contract violations, resource utilization and contract rejections as system metrics independent of the economic metrics.

Violations. This metric measures the ratio between the jobs that have been admitted to those that meet their deadlines. *Violations* $= \frac{j_s}{j_a}$, where j_s is the number of jobs that successfully meet their deadlines, and j_a is the number of jobs that are admitted.

Utilization. Utilization is calculated as the average work units processed per second, compared to a theoretical maximum utilization if no resources were idle at any point during the experiment run. *Utilization* $= \frac{w}{w^*}$, where w is the number of work units completed in the experiment, and w^* is the theoretical maximum.

Rejections The rejection metric is calculated as the ratio of jobs in the experiment to jobs being rejected at admission time. *Rejections* $= \frac{j_r}{j_t}$, where j_r is the number of jobs being rejected, and j_t is the total number of jobs.

Demand Correlation. As a measure of global fairness, we consider the correlation of the demand and the price, seen as cost by the user and profit by the provider. Because the *best effort* model does not reject jobs at admission time with our experimental workloads, it is used as as an approximation for demand. *Demand Correlation* $= \rho_{X,Y}$, where $\rho_{X,Y}$ is the Pearson correlation coefficient of the random variables X and Y, X is the time series of 5min averages of the price in the *best effort* run, and Y is the corresponding time series of the admission mechanism being measured (SAC, or CAC).

Efficiency. As an aid to measuring economic efficiency, the utility function quantifies the payoff a user gets from running a job. The first part of the utility function represents users who do not get any payoff unless the job is fully completed. We call this the *inelastic utility*.

$$U_i(w_d, p, v) = \begin{cases} v - p & \text{if } w_d \geq w \\ -p & \text{if } w_d < w \end{cases} \quad (7)$$

where, w_d is the amount of work completed at the deadline, p is the price paid for running the job, v is the valuation of the job (money bid on job), and w is the work requested in the performance contract. The second part of the utility function represents users who get an incremental payoff based on how much work they completed. We call this the *elastic utility*.

$$U_e(w_d, p, v) = min(w_d/w, 1)v - p \quad (8)$$

A weighted linear combination of (7) and (8) is then the final utility function

$$U = \alpha U_i + (1 - \alpha)U_e \quad (9)$$

Taking the sum of all utilities across all jobs, j, gives us a metric of the *social welfare*, which compared to the optimal utility (with off-line knowledge) can be seen as the economic efficiency of the system. *Efficiency* $= \frac{1}{J}\sum_j^J \frac{U(j)}{U^*(j)}$, where U^* is a theoretical optimal utility obtained from an off-line admission control strategy with no charges applied to (9). More elaborate utility functions could have been used, such as time-decaying utility, or any other quasi-linear utility transformation. We did run some experiments with time-decaying utility as well but found that it complicated the model without providing additional qualitative results. Our main point is to show the difference between elastic and inelastic applications with different market models, and we found the above functions to capture this aspect in the simplest and most accurate way.

To determine the reliability of the results, we perform a statistical test based on the mean and the standard deviation of the metrics. Since only 5 repetitions (experiment samples) can be used because of time constraints the standard z-test (e.g $[\mu - 1.96\sigma, \mu + 1.96\sigma]$ as the 5 per cent significance confidence bound) cannot be performed and we have to resort to the pairwise t-student test with sampled standard deviation.

4.3. Results

All the graphs presented in this section show the average across the five experiment repetitions, with one standard deviation marked with error bars. Non-overlapping error bars thus give a visual indication of a significant result.

System Metrics. Figure 1 shows that the BE model violates a large portion of the contracts as a result of heavy load. SAC addresses this problem but still causes a sizeable portion of violations, whereas CAC has virtually no violations (a small portion of violations may still occur because of our choice to base the admission test on the overall performance of a job as opposed to on a per node basis). The Pareto and Normal value distributions cause more violations for both BE and SAC compared to when all jobs are valued equally. CAC on the other hand is less sensitive to value distribution. Contrasting this result with Figure 1(b), we cannot see any clear evidence that the utilization is significantly effected neither by the value distributions, the load, nor the admission model used. However, the variation in utilization is higher with a Pareto value distribution under high load than with other experiment setups. Further, the rejection ratios in Figure 1(c) cannot fully explain the significant difference in contract violations. For example, in the high-load Pareto value distribution experiment 37 per cent of the jobs were rejected with CAC, compared to none for BE, but 61 per cent of the contracts were violated with BE, compared to 1 per cent with CAC. It is also important to point out that both CAC and SAC reject fewer contracts under low load, which proves that the admission decisions adapt to the current demand.

A quantitative summary of the significance of these results is presented in Table 2. It shows a pairwise t-student test at a 5 per cent significant level, and four degrees of freedom (critical t-value 2.1318) with the null hypothesis that the mean value of the metrics (violations and utilization) are the same. A rejection of the null hypothesis (a significant difference exists) is marked in bold.

Based on these results the main conclusion is that SAC and CAC improved contract fulfillment significantly without causing significant underutilization.

Economic Metrics. Turning to the economic metrics of fairness and efficiency, we can see in Figure 1(d) that the demand correlation coefficient is high for both the SAC and CAC models (about $0.6 - 0.8$ for all experiments). This result indicates that the admission control models are not biased towards the consumer nor the provider, in their pricing structure, and can thus be considered globally fair. The effi-

ciency graphs in Figure 2 show the sensitivity to the elastic versus inelastic utility models. Typically a consumer with a completely inelastic ($\alpha = 1$) utility would choose the CAC or possibly the SAC model. We see that the admission models showed no significant difference in efficiency under low load. However, when there is high contention for the resources and there is high variability in the valuation of jobs (Pareto or Norm), the BE model efficiency deteriorates significantly with an increasing portion of inelastic utility. SAC does not suffer as much from efficiency loss as BE, but it has the same decreasing trend, whereas CAC is completely independent of the utility model chosen (As seen by the straight lines in the graphs). We further note that the efficiency of the SAC and BE models do not decline as much when all jobs have equal value, compared to when they have Pareto or Normal value distributions.

The utility function used to calculate efficiency benefits high-valued jobs with low cost. Comparing Figure 2(a) with Figure 2(b), we can see that the CAC model and to some extent also the SAC model maintained a high level of service for high-valued jobs when the system went from low to high resource contention, a typical trait of a good admission control model.

As previously mentioned, lower utilization and fewer contract violations may be explained as an inherent result of more rejections, as opposed to a feature of the admission control model. To study to what extent the behavior of the SAC and CAC models can be explained by this inherent effect, we modify the SAC model to just randomly reject as many requests as in the Pareto high-load CAC scenario (about 37 per cent). If we first compare the CAC model to the random admission control model (RAC) in Figure 3, we see that the contract violations are considerably higher with the random model although the number of rejections are the same. This shows that the CAC model makes better decisions regarding which jobs to reject. The utilization and the demand correlation (market fairness) are however similar. Looking at the efficiency it is clear that the random model, which does not consider the valuation of jobs, performs worse both than the original SAC implementation and CAC, for both elastic and inelastic jobs. We thus conclude that the inherent effect of rejections cannot explain the contract violation or efficiency properties of the admission control models that we have implemented. We note that it is customary to investigate equilibria of the system when making claims about efficiency in economics. In an experiment which mimics real usage, with randomly configured batch jobs entering and leaving the system continuously, it is very hard to observe and reproduce such stable states. As an alternative we derived our claims from statistically stable states using techniques such as the t-student test. In the following section we will complement the statistical verification with an analytical verification of the results.

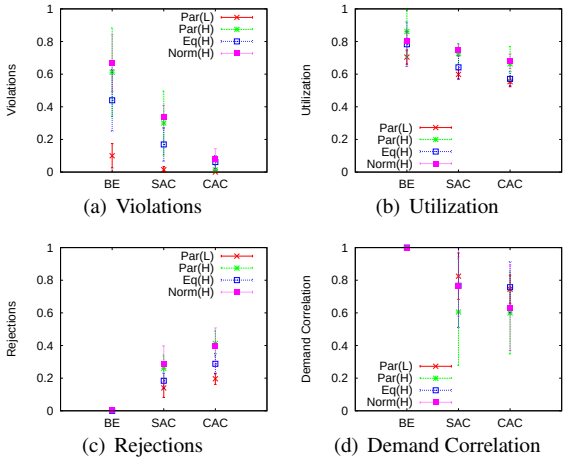


Figure 1. Violations, Utilization, Rejections and Demand Correlation with different Value Distributions (Pareto/Equal/Normal) and Load (High/Low)

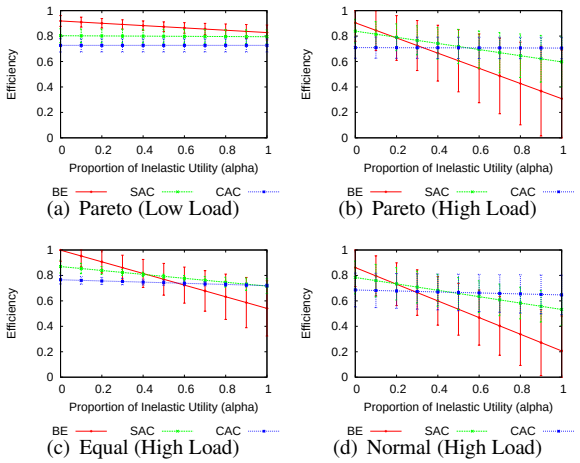
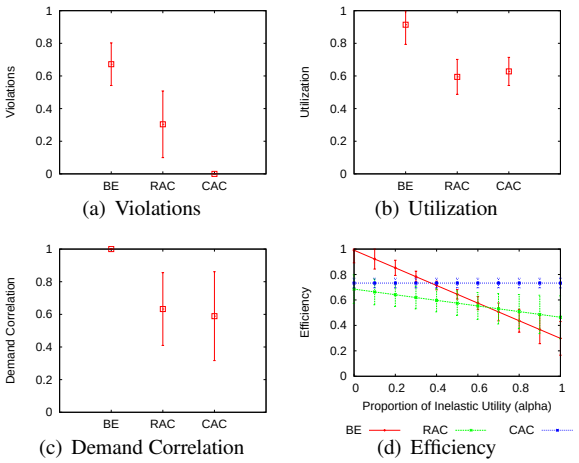


Figure 2. Efficiency with different Value Distributions and Load

Table 2. T-student test (t-values) of mean-value difference for contract violations and utilization. Significant differences (at a 5% level) are marked in bold. Positive values indicate that the mean of the first series compared is higher.

BE/SAC	Par(L)	Par(H)	Equal	Norm
Violations	**2.48**	2.08	**2.82**	**3.94**
Utilization	**4.52**	1.67	2.06	1.79
BE/CAC	Par(L)	Par(H)	Equal	Norm
Violations	**2.99**	**4.90**	**4.39**	**7.09**
Utilization	**6.30**	2.05	**3.33**	**3.74**
SAC/CAC	Par(L)	Par(H)	Equal	Norm
Violations	1.71	**3.25**	**2.20**	**5.91**
Utilization	**2.35**	0.72	1.89	**2.60**



(a) Violations (b) Utilization (c) Demand Correlation (d) Efficiency

Figure 3. Random Admission Control compared to Best Effort and Capacity Admission Control

5. Results Analysis

First, we explain the difference in efficiency between admission control and best effort with inelastic utility analytically. Let $p_w \in [0,1], p_a \in [0,1]$, be the probability of fulfilling the contract, and the probability of admitting a job, respectively. Using the inelastic utility function in (7) we obtain $U = p_a(p_w(v-c) - (1-p_w)c)$. We now study two cases, U_b where all requests are admitted at the expense of some not fulfilling their contracts (e.g. best effort), i.e. $p_a = 1$, and U_g where all jobs fulfill their contract at the expense of some being rejected (e.g. capacity admission control) , i.e. $p_w = 1$. Now by comparing

$$U(p_a = 1) = U_b = p_w(v-c) - c(1-p_w) = p_w v - c \quad (10)$$

and

$$U(p_w = 1) = U_g = p_a(v-c) \quad (11)$$

we get

$$U_g > U_b \Leftrightarrow p_a v - p_a c - p_w v + c > 0 \quad (12)$$

Setting $q = c/v$ we have

$$U_g > U_b \Leftrightarrow p_a > \frac{p_w - q}{(1-q)}. \quad (13)$$

Using the experimental values of the runs from Figure 2 we for instance have for
(a):

$$0.80 < \frac{0.90 - 0.10}{1 - 0.10} \approx 0.89 \Rightarrow U_b > U_g \quad (14)$$

(b):

$$0.63 > \frac{0.39 - 0.14}{1 - 0.14} \approx 0.29 \Rightarrow U_g > U_b \quad (15)$$

(c):

$$0.71 > \frac{0.56 - 0.12}{1 - 0.12} = 0.5 \Rightarrow U_g > U_b \quad (16)$$

and (d):

$$0.60 > \frac{0.33 - 0.10}{1 - 0.10} \approx 0.26 \Rightarrow U_g > U_b. \quad (17)$$

This analytical result explains and mimics the relative positions of the CAC and BE curves in Figure 2 very well for high values of α (inelastic utility). With a similar line of reasoning and setting $p_d = w_d/w$ (work completion ratio), we can show that

$$U_g > U_b \Leftrightarrow p_a > \frac{p_d - q}{(1-q)} \quad (18)$$

when using the elastic utility function in (8).

To summarize the analytical and experimental results, poor work completion (low values of p_w and p_d) and high cost (high values of q) contribute to a higher efficiency of the admission control strategies compared to the best effort strategy if the number of rejections are kept low (high p_a), which corresponds well to intuition.

Based on these results a user with an inelastic job could compare the cost for a resource on a reservation market (SAC or CAC) and on a spot market (BE) to the valuation of the job to get q, and calculate or estimate the likelihood of fulfilling the contract on the spot market (p_w) versus the likelihood of getting rejected on the reservation market (p_a). The two probabilities may be measured by the user, made available by a 3^{rd} party monitoring service or the provider itself. Now if $p_a > \frac{p_w - q}{(1-q)}$ submitting the job to the reservation market is more efficient.

Conversely, based on historical data, the provider could map the current demand, approximated by the measured IAT, to values of p_a and p_w. A provider that would like to optimize the efficiency without any bias towards elastic or inelastic utility users could evaluate: $p_a > \frac{p_w + pd - 2q}{2(1-q)}$. If it evaluates to false, the provider might want to increase the spot-market partition compared to the reservation partition. Whether to use SAC or CAC guarantees largely depends on the level of control the admission broker has over the resource allocator, but also on the frequency of high contention periods (when CAC tends to outperform SAC). Furthermore, the provider could evaluate how much should be charged (q or c/v) for the capacity admission control service given a certain system state (represented by p_w, p_a and p_d).

6. Related Work

Related work fall into three broad categories, each discussed in turn below, market-based resource scheduling and

allocation, IP traffic engineering, and QoS enabled web servers.

Chun and Culler [4] present a performance analysis of three different scheduling algorithms, FirstPrice (priorities paid for on centralized auction market), SJF (short jobs have priority), and PrioFIFO (three priority queues with different prices set statically) based on aggregate user utility. FirstPrice outperforms both SJF and PrioFIFO significantly for highly parallel jobs. PrioFIFO was sensitive to changes in demand and deteriorated in performance if the wait time in the most expensive queue was long. Our work differs from this work in that our results are independent of which scheduling algorithm is used, our workload is constructed by carefully modeling real traces, and our underlying allocation mechanism is a continuously cleared decentralized spot market auction. These differences are also apparent in extensions of Chun's and Culler's work [14, 22, 1] that study more elaborate scheduling algorithms and utility functions that take resource price and provider profit into account. The combination of reservation and spot market pricing with statistical guarantees is novel and sets this work apart from other microeconomic systems that control job performance in shared clusters for parallel jobs, such as [27, 26, 29, 6, 28, 15, 5].

There is a substantial body of work on Internet Protocol quality-of-service enforcement or traffic engineering, represented by the two IETF specifications IntServ [3], and DiffServ [2]. The IntServ specification takes the approach of reserving paths for individual users, and thus does not scale as well as the DiffServ approach, which is based on marking individual packets with different *per-hop behaviors* in a stateless and decentralized architecture. We are facing the same issues and tradeoffs when allocating computational resources across large distributed systems. However, new virtualization technology and the fact that many of the resources are localized (e.g. CPU, memory, disk) makes it worth revisiting the reservation concepts. Knightly and Shroff [17] provide an evaluation of the different admission control algorithms available for IP traffic shaping. The dilemma of choosing between denying access to flows that might have been served and thereby cause underutilization and serving requests that might break existing QoS contracts makes it hard to use coarse statistical bounds and too simplified assumptions about traffic flow distributions. Put differently, both accuracy maximization and risk minimization are desired. Again, our admission control decision differs from the IP flow one, in that we can, through virtualization, more directly enforce that an admitted request stays within its bounds. Our decision is thus more about making sure that the provider does not lose out on utilization or profit by admitting low priority tasks prematurely.

Admission control as a means to avoid service degradation of high priority tasks during overload has also been extensively studied in the context of Web servers, as exemplified in [10, 7, 19]. Priorities of individual requests are either set explicitly in the server configuration or inferred implicitly by the admission algorithm. Our admission controller, on the other hand, gives users an incentive to specify the priority truthfully themselves. Another key difference is that, in a Web server context, the focus is on optimizing throughput and response time by applying queuing and control theory, and estimating expected service time. Our system does not use centralized queues and the service times are not estimated but explicitly requested by the users, as they can vary greatly. In general, our approach of enforcing contracts by means of resource virtualization provides much finer-grained control over service-levels than purely statistical load estimates.

7. Conclusions

Using a statistically accurate, representative and realistic workload model, we have experimentally shown how a simple statistical admission control mechanism can improve contract fulfillment without causing underutilizition during times with high resource contention. Based on two intuitive utility functions, elastic and inelastic, we have also shown that the system remains efficient, i.e. exhibits high social welfare or aggregate utility, even under heavy contention and with various job valuation distributions, such as Equal, Pareto and Normal. The efficiency results were analytically verified and constraints on resource cost, admission ratio, and contract violation ratio were derived to inform which admission policy is best for different utility functions given a certain system state.

Acknowledgments

We would like to thank our colleagues and collaborators, Bernardo Huberman, Tad Hogg, Li Zhang, Lars Rasmusson and John Wilkes for lucid discussions. Thanks also to Cynthia Bailey Lee for clarifying and explaining patterns seen in the SDSC workload trace.

References

[1] A. AuYoung, L. Grit, J. Wiener, and J. Wilkes. Service contracts and aggregate utility functions. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing (HPDC)*, June 2006.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, IETF, December 1998.

[3] R. Braden, S. Clark, and S. Shenker. Integrated services in the internet architecture. RFC 1633, IETF, June 1994.

[4] Brent N. Chun and David E. Culler. User-centric Performance Analysis of Market-based Cluster Batch Schedulers. In *Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid*, 2002.

[5] Brent N. Chun and Philip Buonadonna and Alvin AuYoung and Chaki Ng and David C. Parkes and Jeffrey Shneidman and Alex C. Snoeren and Amin Vahdat. Mirage: A Microeconomic Resource Allocation System for SensorNet Testbeds. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors*, 2005.

[6] R. Buyya, M. Murshed, D. Abramson, and S. Venugopal. Scheduling Parameter Sweep Applications on Global Grids: A Deadline and Budget Constrained Cost-Time Optimisation Algorithm. *Software: Practice and Experience (SPE) Journal*, 35(5):491–512, April 2005.

[7] X. Chen, P. Mohapatra, and H. Chen. An admission control scheme for predictable server response time for web accesses. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 545–554, New York, NY, USA, 2001. ACM Press.

[8] 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.

[9] B. Efron and R. J. Tibshirani. *An Introductin to the Bootstrap*. Chapman and Hall, 1993.

[10] S. Elnikety, E. Nahum, J. Tracey, and W. Zwaenepoel. A method for transparent admission control and request scheduling in e-commerce web sites. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 276–286, New York, NY, USA, 2004. ACM Press.

[11] D. G. Feitelson. Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/, 2007.

[12] D. Ferguson, Y. Yemimi, and C. Nikolaou. Microeconomic Algorithms for Load Balancing in Distributed Computer Systems. In *International Conference on Distributed Computer Systems*, pages 491–499, 1988.

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

[14] D. Irwin, J. Chase, and L. Grit. Balancing Risk and Reward in Market-Based Task Scheduling. In *International Symposium on High Performance Distributed Computing*, 2004.

[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] S. D. Kleban and S. H. Clearwater. Quelling queue storms. In *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, page 162, Washington, DC, USA, 2003. IEEE Computer Society.

[17] E. W. Knightly and N. Shroff. Admission control for statistical qos: Theory and practice. *ieeenet*, 13(2):20–29, March/April 1999.

[18] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang, and B. A. Huberman. Tycoon: an Implemention of a Distributed Market-Based Resource Allocation System. *Multiagent and Grid Systems*, 1(3):169–182, Aug. 2005.

[19] S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau. Admission control and dynamic adaptation for a proportional-delay diffserv-enabled web server. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 172–182, New York, NY, USA, 2002. ACM Press.

[20] 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.

[21] M. E. Pate-Cornell. Uncertainties in risk analysis: Six levels of treatment. *Reliability Engineering and System Safety*, 54(2):95–111, 1996.

[22] F. I. Popovici and J. Wilkes. Profitable services in an uncertain world. In *SC05: Proceedings of Supercomputing*, 2005.

[23] 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.

[24] T. Sandholm and K. Lai. A Statistical Approach to Risk Mitigation in Computational Markets. In *Proceedings of the ACM International Symposium on High Performance Distributed Computing (HPDC)*, June 2007.

[25] T. Sandholm and K. Lai. Prediction-based enforcement of performance contracts. In *GECON '07: Proceedings of the 4th International Workshop on Grid Economics and Business Models*, 2007.

[26] I. Stoica, H. Abdel-Wahab, and A. Pothen. A Microeconomic Scheduler for Parallel Computers. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 122–135, April 1995.

[27] 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.

[28] 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.

[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.