User-centric Performance Analysis of Market-based Cluster Batch Schedulers

Brent N. Chun and David E. Culler

University of California at Berkeley
Computer Science Division
{bnc,culler}@cs.berkeley.edu
http://www.cs.berkeley.edu/~{bnc,culler}

Abstract

This paper presents a performance analysis of market-based batch schedulers for clusters of workstations. In contrast to previous work, we use user-centric performance metrics as the basis for system evaluation. Each user is modeled as having a utility function for each job which measures value delivered to the user as function of execution time. Summing over all utility functions in the workload, we use aggregate utility as a measure of overall value delivered to users. With aggregate utility as the performance metric, simulations are used to quantify the performance of both market-based and traditional batch scheduling algorithms under a variety of synthetic workloads. Results show that an auction-based batch scheduling algorithm improves performance by a factor of up to 2-5x for sequential workloads and up to 14x for highly parallel workloads compared to traditional scheduling algorithms.

1 Introduction

Batch scheduling is a common paradigm for large-scale, production cluster computing environments. Users submit jobs in batch to a queue and a centralized scheduler decides how to prioritize them and submit them for execution on cluster nodes. Scheduling algorithms in these systems must decide how best to prioritize competing jobs of varying levels of importance and how to manage queue lengths so that response time does not become excessively large. Effectively performing these tasks requires knowledge of how users value the resources being competed for and having a feedback signal that prevents users from submitting unbounded amounts of work. Unfortunately, current approaches to batch scheduling provide little, if any, means for users to express resource valuations and influence job queue priorities. In addition, while feedback signals are provided, there are often no associated incentives for users to pay attention to and respond to them.

To address the shortcomings of traditional batch systems, we advocate use of market-based resource management [3, 4, 6, 8, 10]. With market-based resource management, computational resources are allocated through markets where the cost of using a resource is directly related to supply and demand. Resource allocations are determined through use of economic mechanisms such as auctions where users place explicit valuations on the resources being contended for as expressed by mechanism-specific "bids". Explicit valuations provide systems with extra information which is used to optimize for user value as opposed to system-centric performance metrics such as mean slowdown. Pricing combined with charging creates feedback which causes users to balance the amount of work submitted to the system with the cost of obtaining associated resources. We hypothesize that explicit resource valuations and price feedback can lead to substantially higher value delivered to users.

Previous work in market-based batch scheduling [5, 7, 10] has mainly used auctions as the economic mechanism for resource allocation. Users assign bids to jobs which reflect importance and bids are used in an auction to determine job priority in the queue. Compared to these efforts, our work differs most notably in our use of *user-centric performance metrics* as the basis for system evaluation. In our analysis, each user is modeled as having a utility function for each job which measures value delivered to the user as function of execution time. User-centric performance metrics are used which focus on user value as opposed system-centric metrics which do not take utility into account and thus are not good measures of how satisfied users were with their resource allocations. To measure overall system performance in a user-centric manner, all users should be taken into account and value delivered to users should be the basis of performance. In this paper, we use aggregate utility to quantify overall value delivered to users.

The rest of this paper is organized as follows. In Section 2, we provide an overview of cluster batch systems and describe the three batch schedulers we compare in our analysis. In Section 3, we describe our methodology for performance evaluation and make a case for user-centric performance metrics as the basis for system evaluation. In Section 4, we present our simulation results which quantify the benefits of market-based systems over traditional approaches. Section 5 describes related work. Finally, Section 6 concludes the paper.

2 Cluster Batch Schedulers

Batch systems are an essential part of production cluster computing environments. With user resource demands large enough to outstrip resource capacity on even the largest of clusters, batch systems are important because they allow incoming work to be assigned to cluster resources without overloading the system. In batch systems, users submit jobs in batch and jobs are subsequently queued for execution by a batch scheduler. As resources become available, jobs from the queue are scheduled onto free resources according to a scheduling algorithm which determines how jobs in the queue are ordered for execution based on optimization of some performance metric. In this section, we provide an overview of the three main approaches to batch scheduling on clusters, highlight some of the problems with existing approaches, and provide the motivation for market-based cluster batch scheduling.

2.1 Scheduling Algorithms

Traditional batch scheduling algorithms prioritize the queue based on optimization of system-centric performance metrics such as mean response time. In doing so, they afford users little or no control over how their jobs are given priority in the queue. FIFO schedulers, for example, order jobs based on arrival time and make no effort take the importance of jobs into account. SJF, another common policy, does not do much better. Jobs are ordered by CPU lengths to minimize mean slowdown, but this algorithm is suboptimal if job lengths are not directly correlated with how users value the resources. Schemes that use multiple FIFO or SJF queues with different scheduling priorities are also ineffective without associated incentives controlling their use. Without additional incentives, there is nothing to prevent every user from requesting service from the highest priority queue.

Supercomputing centers augment traditional batch scheduling algorithms with charging. In their most flexible configuration, batch scheduling is implemented through a set of FIFO queues each with a different scheduling priority. Users are free to assign their jobs to any of the queues. When scheduling jobs on free resources, the batch scheduler chooses the earliest job in the highest priority queue that non-empty. Each FIFO queue also has a charging rate associated with it that is related to its priority. Higher priority queues charge higher rates for computational resources when jobs eventually execute. These charges create incentives which prevent users from always assigning their jobs to the highest priority queue. This policy, which we refer to as PRIOFIFO, thus provides users with a way to express coarse-grain, static valuations which influence how soon their jobs are scheduled.

Market-based batch schedulers described in the literature [5, 7] have used auctions to determine the most eligible job. With auction-based scheduling, users assign bids to jobs which reflect their valuations of the underlying resources (i.e., important jobs have higher valuations). The batch scheduler in turn uses an auction to order the jobs in the queue based on the amount per unit of CPU time the user is willing to pay. The most eligible job is the one willing to pay the highest rate per unit of CPU time. Because users can assign arbitrary bids, valuations can be expressed at a very fine granularity, and because charging is associated with resource usage, incentives exist to prevent users from always assigning the highest possible bid. In this paper, we examine a market-based algorithm, FIRSTPRICE, that uses a first price auction for scheduling and allows users to assign *time-varying* bids which determine job priority in the queue. Charging along with finite, periodically-funded bank accounts prevent users from assigning arbitrarily high bids.

3 Methodology

Our methodology for system evaluation is based on simulation, synthetic workloads and resource valuations, and evaluation based on user-centric performance metrics. We use simulation for performance analysis since it allows us to examine the widest possible range of systems and workloads. We use synthetic workloads and valuations in order to examine performance sensitivity to variations in workload parameters and how users value computational resources. Finally, we evaluate the performance of both market-based and traditional systems using user-centric performance metrics as opposed to system-centric metrics such as mean slowdown. We use user-centric performance metrics because traditional performance metrics are not consistent with total value delivered to users.

3.1 User-centric Performance Metrics

All users have some notion of value when running applications on a cluster. A user simulating a cure for cancer, for example, places higher value on those simulations than, say, image processing jobs to create new images for a web page. The importance of a job can viewed as a time-dependent valuation of the resources being contended for. These valuations have both a magnitude, which expresses importance, and a rate of decay which reflects user sensitivity to delay. Important jobs have higher valuations while less important jobs have lower valuations. Intuitively, an optimal batch scheduling policy ought to take these valuations into account when assigning priority to jobs in the queue. Unfortunately, traditional systems provide little, if any, means for users to express resource valuations and influence queue priorities, and thus queue priority is rarely aligned with user valuations of the underlying resources.

Market-based systems make resource valuations explicit by allowing users to express resource valuations to the system and influence their resource allocations. Using this extra information, market-based batch systems optimize for user value by assigning queue priorities based on how much users value the resources. In this paper, we assume valuations are piecewise-linear functions which decay linearly over time as a function of slowdown (Figure 1). Value is assumed to be constant up to a slowdown of 1 since users cannot expect a job to finish sooner than it would with dedicated access. Value then decays linearly until slowdown reaches some critical value, at which point value delivered is 0 regardless of completion time. We assume users have some tolerance for delays due to queueing delays which is modeled by the slope of linear decay. We call the length of time when value begins to decay to the time when value decays to 0 the user's *delay tolerance* and measure it as a multiple of job length.

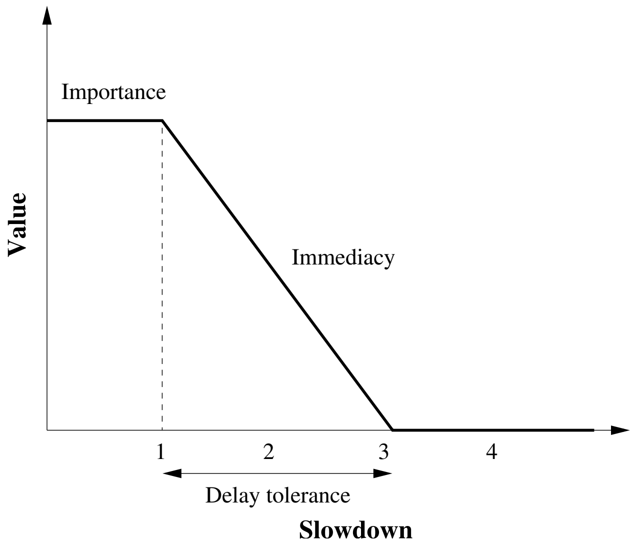


Figure 1: **Piecewise-linear valuation.** Value delivered to the user as a function of job slowdown.

In contrast to previous work, we use user-centric performance metrics to evaluate system performance. As previously described, all users have some notion of value associated with each job submitted to the system. Given a common medium of expression (e.g., currency in a market-based system), value can be expressed concretely and valuations can be compared in a uniform manner for scheduling purposes. To measure overall system performance in a user-centric manner, all users should be taken into account and value delivered to users should be the basis of performance. In this paper, we use aggregate utility as a measure of overall value delivered to users. We use aggregate utility as the metric since traditional performance metrics do not explicitly take value into account and thus are not accurate measures of total value delivered to users. We compute aggregate utility by taking the value delivered for each job, as computed by the corresponding user's valuation, and summing over all jobs in the workload.

We use the sum of the valuations to compute aggregate utility for both traditional and market-based systems. In both cases, using the sum of the valuations is appropriate since the bottom line for delivering value to users is by definition the sum of the valuations. Using the sum of the valuations also allows us to make meaningful comparisons between the two types of systems. To support these types of comparisons, the economist's view of periodic funding of user bank accounts (positive utility) and costs associated resource consumption (negative utility) are not factored into performance when evaluating market-based systems. Instead, we assume that cost only plays a role in determining user behavior. For example, when auctions are used to allocate resources, users take cost into account when assigning their bids and do not assign high bids to low valued jobs.

In the market-based systems we consider, the use of funds is solely a means to ultimately deliver more value to users. Funds have no intrinsic value and are only useful in exchange for computational resources on a single cluster. Because of this, it makes no sense to factor periodic funding or costs into performance for market-based systems. If these variables were factored into performance, then it would be possible that a market-based system could "perform" better or worse than a traditional system even when processing the same workload in an identical fashion. In addition, resource management policies that artificially delay execution, while obtaining more periodic funding, would also appear to be delivering better performance when in fact they are actually delivering less value to users. To avoid these anomalous results, we use cost solely for modeling user behavior and assume that users always assign bids based on true valuations.

3.2 Simulation Configuration

We have written a simulator called Stingray for flexible simulation of various cluster resource management systems. Using Stingray, we have conducted a series of performance sensitivity experiments comparing the performance of an auction-based scheduling algorithm with two commonly used traditional scheduling algorithms. The system consisted of a 32 node cluster with single processor nodes. Workloads consisted of 32 users submitting batches (i.e., bursts) of sequential and parallel jobs with normally distributed burst interarrival times, normally distributed job CPU lengths, and either fixed (1 in the sequential case) or uniformly distributed job parallel degrees. Valuations were assumed to be bimodal where 80% of the valuations come from a low normal distribution (low value jobs) and 20% of the valuations come from a high normal distribution (high value jobs). High valued jobs were assumed to be, on average, 100x more important than low value jobs. Modeling valuations as bimodal seemed natural and we feel is reasonable assumption for many workloads. Finally, since cluster workloads are often demanding, we also examined sensitivity to burstiness in the workload and various levels of system utilization.

4 Performance Analysis

In this section, we analyze the performance of a market-based batch queue system in terms of aggregate utility while comparing it to two commonly used traditional algorithms. We test our hypothesis that by allowing users to express resource valuations that significantly more value can be delivered to users. We test it by performing a series of experiments which measure performance sensitivity to workload parameters and valuation distributions including burstiness, utilization, parallelism in the workload, and user sensitivity to delay. With the base performance established, we then extend these results by examining what performance gains can be achieved through use of preemption. Since scheduling decisions are not always optimal due to the lack of future information, these experiments quantify the performance gains of being able to correct previous scheduling decisions which may no longer be optimal in light of new information.

We analyze three different algorithms for these experiments: SJF (shortest job first), PRIOFIFO (i.e., supercomputing center policy), and FIRSTPRICE, a market-based algorithm. Each algorithm imposes an ordering on how queued jobs are scheduled onto the cluster. For SJF, it is based on smaller job CPU lengths. For PRIOFIFO, it is based on queue priority and arrival times. With PRIOFIFO, users have their choice of submitting jobs to one of three FIFO queues each with a different priority and charging rate [1]. The next job to be scheduled is always the earliest job in the highest priority queue that is non-empty. Finally, for FIRSTPRICE, queue ordering is based on user-assigned, time-varying bids which specify value per unit time. All three algorithms implement backfilling which allows jobs farther back in the queue to be scheduled onto idle nodes while the most eligible parallel job waits for all its nodes to become available. (In no case does a backfilled job delay the execution of the most eligible job in the queue.) In all simulations, the batch scheduling algorithm is run at periodic intervals, and at each interval as many jobs as possible are scheduled onto available resources.

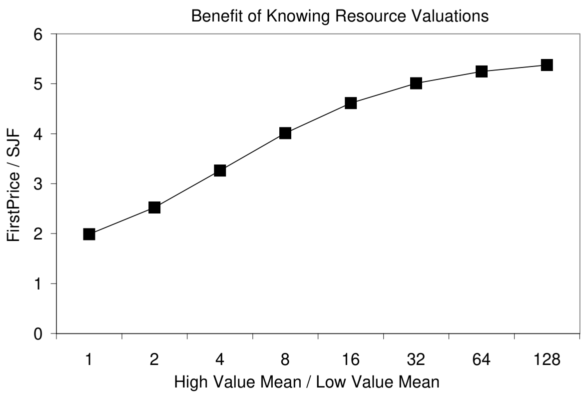
4.1 Explicit Resource Valuations

In these experiments, we quantify the value of having explicit resource valuations determine scheduling priority in the queue. We do this by comparing the performance of a market-based algorithm with explicit valuations (FIRSTPRICE) with a traditional algorithm without explicit valuations (SJF). We conduct five experiments using synthetic workloads and valuation distributions, each examining performance sensitivity to one particular aspect of the workload. More specifically, we analyze performance sensitivity to variations in resource valuations, user sensitivity to delay, burstiness of job arrivals, system utilization, and parallel in the workloads. The results from these experiments demonstrate under what types of workloads and in which regimes of operation market-based systems are most effective compared to traditional algorithms. Figure 2 plots the results.

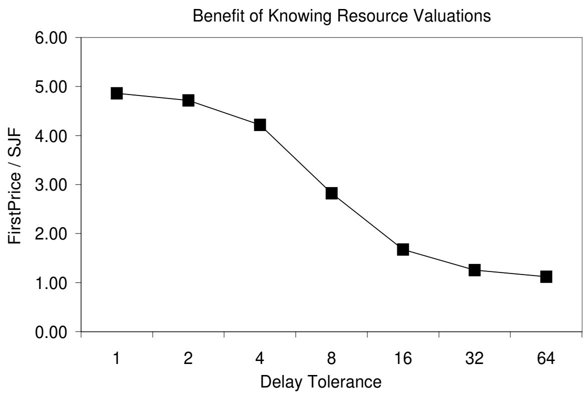
The results are consistent with intuition. The more variation in resource valuations, the more opportunities FIRSTPRICE has to reorder the queue based on the valuations and so performance should improve with the differences in valuations. The more sensitive users are to delay, the larger the benefit of being able to decrease queueing delays for important jobs through assignment of high valuations. As users become more sensitivity to delay, market-based systems should be more effective. The burstier the workload, the longer the queues and the more competition there is for scarce resources. As burstiness increases, performance should also increase since scheduling based on valuations allows the system to focus on the jobs that are delivering the most value. As system utilization increases, so should the the probability of competition and so performance should also increase. Finally, with more parallelism in the workload, the more likely the most eligible job will block waiting for all its required nodes to become free. Head-of-line blocking increases queueing delays, creates longer queues, and thus presents more opportunities for market-based systems to reorder the queue.

Overall, we observe performance improvements of up to 2-5x for sequential workloads and up to 14x for highly parallel workloads when resource valuations are known. These substantial performance gains can be directly at-

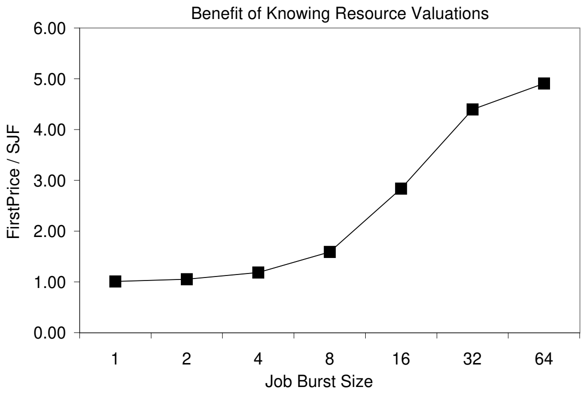
[1] We optimistically assume that prices are appropriately set so that low valued jobs map into the lowest priority queue and high valued jobs map into the highest priority queue. Assuming valuation distributions do not change much over time, such prices might be set based on empirical observations of usage of the different queues.



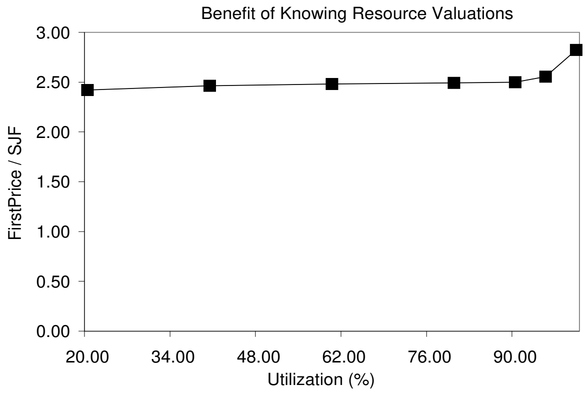
(a)



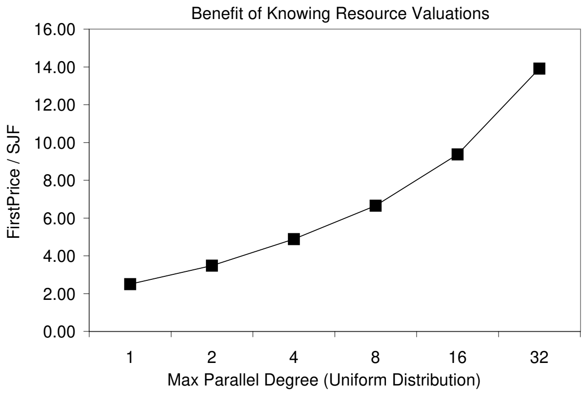
(b)



(c)

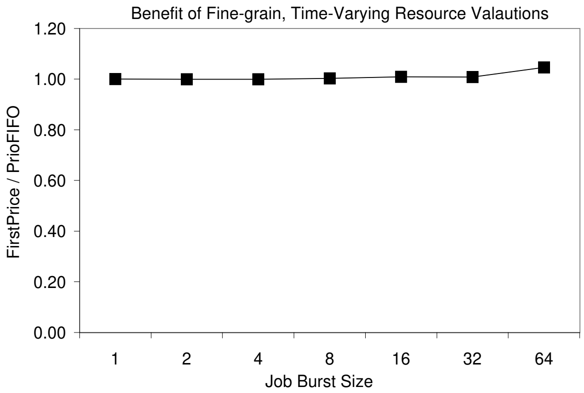


(d)

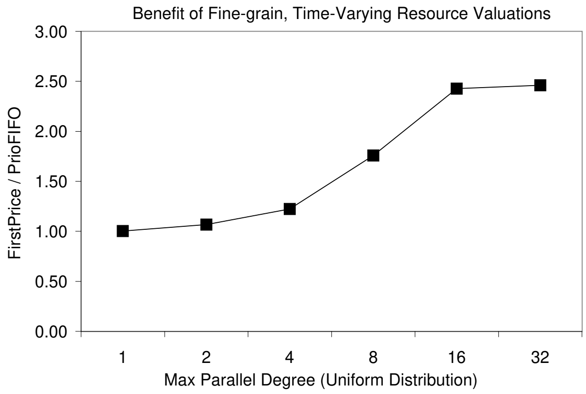


(e)

Figure 2: **Benefit of knowing resource valuations.** This figure compares aggregate utility delivered to users when valuations are known (FIRSTPRICE) versus when they are not known (SJF). For sequential workloads, we observe that the benefits of knowing valuations are most pronounced when workloads are bursty (c.) and when there is significant variation in resource valuations (a.). In addition, we also see that as users become more sensitive to delay (b.) that the benefit of scheduling based on valuations increases substantially. For even moderately bursty workloads, significant benefits are obtained over a wide range of utilization (d.) due to competition in the queue. Finally, we see that as workloads have more parallelism (e.), the benefits of knowing valuations increases significantly.



(a) Sequential jobs



(b) Parallel jobs

Figure 3: **Benefit of fine-grain, time-varying resource valuations.** This figure compares aggregate utility delivered to users when valuations are fine-grain and time-varying (FIRSTPRICE) versus when they are coarse-grain and static (PRIOFIFO). For sequential jobs, we observe that coarse-grain, static valuations are just as effective as fine-grain, time-varying valuations for workloads with varying levels of burstiness. (We observed similar results for other sensitivity experiments for sequential jobs which are not shown here.) On the other hand, for parallel jobs, we observe that as parallelism in the workload increases, so do the benefits of having fine-grain, time-varying valuations.

tributed to the use of fine-grain, time-varying resource valuations which determine priority in the batch queue. Fine-grain valuations provide users with a mechanism to prioritize their jobs in the queue relative to other users jobs. Charging combined with auction-based scheduling provides incentives which prevent users from assigning arbitrary high bids and instead bidding based on their true valuations of the underlying resources. Fine-grain valuations alone account for the majority of the performance improvements. In addition, time-varying bids provide an additional way for market-based batch schedulers to improve performance. By allowing assignment of time-varying bids, the batch scheduler always has a consistent view of the current value of all jobs in the queue. When long queues develop, valuations of the jobs in the queue change due to different user tolerances to delay. Without time-varying bids, scheduling would be based on old information which is no longer valid.

4.2 Fine-grain, Time-varying Valuations

In these experiments, we quantify the value of having fine-grain, time-varying valuations (FIRSTPRICE) determine queue priority as opposed to coarse-grain, static valuations (PRIOFIFO). We present results from two experiments which analyze performance for both sequential and parallel workloads. For sequential workloads, we repeat the experiment for explicit valuations by examining performance sensitivity to burstiness in the workload. For parallel workloads, we also repeat a previous experiment by examining performance sensitivity to the degree of parallelism in the workload. Figure 3 plots the results.

The results show that compared to an optimally configured supercomputing center policy of having multiple FIFO queues with different scheduling priorities, a first price auction delivers up to 2.5x higher performance for highly parallel workloads and comparable performance for sequential workloads. For parallel workloads, this result can be attributed to increased head-of-line (HOL) blocking in high priority queues for PRIOFIFO. As parallelism in the workload increases, the probability that the most eligible job in the queue blocks while waiting for all its nodes to become available increases. Increased HOL blocking in turn causes longer queues to form and consequently reduces the effectiveness of the high priority queues as a way for high valued jobs to see small queueing delays. FIFO processing of a long, high priority, queue with jobs of varying levels of importance and sensitivities to delay is suboptimal as queue length increases. On the other hand, for sequential jobs, we observe that coarse-grain, static valuations are just as effective as having fine-grain, time-varying valuations. (We observed similar results for other sensitivity experiments for sequential jobs which are not shown here.) This result can be attributed to smaller queueing delays in the high priority queue for sequential workloads.

Based on our results, we advocate use of first price auctions for batch scheduling of compute-intensive jobs for several reasons. First, we showed that performance using a first price auction is universally better than both SJF and PRIOFIFO under all workloads simulated. Second, the amount of work needed to implement a first price auction is small. In this work, we optimistically assumed that queues in PRIOFIFO were optimally priced so queueing delays in high priority queues were small. In practice, this requires empirically determining prices and potentially repricing queues based on changing demand. The amount of work to implement this is likely to be more than that required to implement a first price auction. Finally, first price auctions are easy for users to reason about. Bids are scalar quantities which represent overall value and charging is simple to understand. Supercomputing centers, on the other hand, typically implement complex charging algorithms which make it difficult for users to get a handle on what is going on.

4.3 Preemption

In this section, we evaluate preemption as a means to correct previous scheduling decisions in light of new information. We analyze preemption because we anticipate batch systems can benefit by preempting low valued jobs to run high valued jobs which had not arrived when the low valued jobs were originally scheduled. To evaluate preemption, we assume preemption occurs whenever an pairwise opportunity to improve performance arises. Preemption occurs whenever swapping the execution order of some running job with the most eligible job in the queue results in increased revenue for the system. Since we assume users always assign bids which reflect their true valuations, this strategy is pairwise optimal in terms of aggregate utility. In all simulations, jobs are preempted at most once in order to help avoid starvation.

Figure 4 plots the results for FIRSTPRICE with and without preemption as a function of user tolerance to delay. Intuition suggests that benefits of preemption should increase with sensitivity to delay since the penalty for waiting becomes larger with longer queueing delays. Surprisingly, we see that only for the most time sensitive of users does preemption actually make a difference and even then the improvements are fairly small (up to 20%). (Other experiments, omitted due to space constraints, also show either little gain, no gain, or in some cases slight reductions in performance.) The main reason for this result is expected queueing delay for high valued jobs. Since scheduling based on resource valuations already provides high valued jobs a way to bypass low valued jobs in the queue, any benefits realized through preemption must provide high valued jobs a way to wait even less in the queue. For the workloads we examine, we observe that such opportunities do not make a significant impact on performance since the expected time for resources to free up is not very significant.

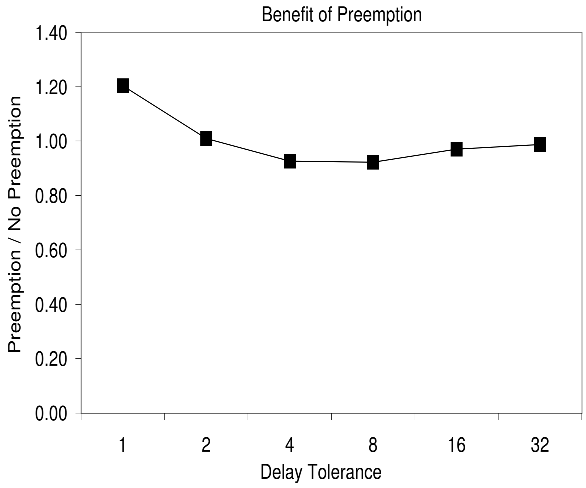


Figure 4: **Benefit of preemption.** This figure compares aggregate utility delivered to users using the FIRSTPRICE algorithm with and without preemption. We observe that preemption in batch environments delivers little or no benefits except when users are highly sensitive to delay where performance improves by at most 20%. Other experiments, omitted due to space constraints, showed similar results in all but very specialized cases.

5 Related Work

Spawn is the earliest known work in applying market-based resource management for batch scheduling in distributed systems [10]. In Spawn, Vickrey auctions are used to provide market-based resource allocation of dedicated CPU time. Workloads are assumed to be tree-based, concurrent applications. Applications have funding rates associated with different nodes of the tree which are used to purchase CPU resources for tasks associated with leaves beneath those nodes. CPU resources are purchased by participating in Vickrey auctions for dedicated slices of CPU time which are held independently by each node. Which auctions to participate in is (presumably) determined randomly since no information is revealed through the Vickrey auction for balancing load. Through simulation and measurement on a prototype implementation, their main result shows that application funding rates are effective in achieving proportional-share resource allocations. While applications were allowed to express valuations through bids, no evaluation was done based on aggregate utility.

Stoica's microeconomic parallel scheduler [7] also takes an auction-based approach using a first price auction for market-based resource allocation of dedicated CPU time for parallel jobs. In his system, jobs are assigned saving accounts which are used to purchase CPU resources. CPU resources are purchased by participating in a centralized first price auction that allocates CPU time on all processors. The winning job is the job with the highest bid per unit time per processor and is scheduled immediately if enough nodes are available. If enough nodes are not available, the winning job blocks and is charged the time it takes until enough free nodes are available. In other words, there is no backfilling and jobs are charged for head-of-line blocking. Through simulation, Stoica's main result is comparable, and in some cases better, mean system and user response times compared to FIFO and FIFO with backfill scheduling policies. He also shows that income rates are inversely proportional to mean response times. Compared to our work, this work differs mainly in its use of traditional performance metrics as opposed to aggregate utility in evaluating system performance.

Geweke's economic batch queue [5] also uses a Vickrey auction for market-based resource allocation of CPU time for parallel jobs on a cluster. In his system, jobs are assigned bids which are used to purchase CPU resources through a centralized Vickrey auction that allocates CPU time on all nodes. Winning jobs are scheduled immediately regardless of node availability through the use of preemption – running jobs are preempted if they are outbid. The system was implemented through modification of the PBS batch queue system and was deployed on a real system. Unfortunately, demand was too limited to draw any significant conclusions. Instead, results were obtained mainly through simulation which essentially confirm previous results [4, 7] that higher bids yield smaller mean system and user response times. Preemption, the most interesting aspect of this system, was not evaluated at all. Compared to our work, this work also differs in its use of system-centric performance metrics as opposed to user-centric performance for system evaluation.

In addition to market-based approaches, there is also a body of work on the evaluation of traditional batch scheduling algorithms for both clusters of workstations and specialized parallel machines (e.g., massively parallel processors) using traditional performance metrics. Representative examples include [9] which evaluates three traditional batch scheduling algorithms (FIFO, SJF, LJF) for clusters of workstations, [1] which evaluates batch schedulers for the NEC Cenju-3 supercomputer, and [2] which evaluates batch systems for the SGI Origin 2000 parallel machine. In all cases, traditional performance metrics such as average waiting time and average response time were used as the basis for system evaluation. However, as already discussed, traditional performance metrics are generally not accurate measures of overall value delivered to users. Thus, while a traditional algorithm may perform optimally based on some system-centric performance metric, the end result will generally be that users are less satisfied with their resulting resource allocations compared to a market-based approach.

6 Conclusion

In this paper, we have quantified the benefits of market-based resource management for batch scheduling on clusters of workstations. We made a case for user-centric performance metrics as the basis for evaluating system performance. We presented aggregate utility as one such metric which measures overall value delivered to users. Using aggregate utility as the performance metric, our simulations showed that using a first price auction for batch scheduling improves performance by a factor of 2-5x for sequential workloads and up to 14x for highly parallel workloads compared to a traditional shortest job first algorithm. We also showed that compared to an optimally configured supercomputing center policy of having multiple FIFO queues with different scheduling priorities, a first price auction delivers up to 2.5x higher performance for highly parallel workloads and comparable performance for sequential workloads. Finally, we showed that given the ability to express valuations and influence queueing priority, preemption does not add significant value.

Our results demonstrate that market-based resource management results in significantly more value delivered to users under a variety of workloads compared to traditional approaches. These substantial performance gains can be directly attributed to the use of fine-grain, time-varying resource valuations which determine priority in the batch queue. We saw that compared to no explicit valuations, market-based systems are a huge improvement since, without valuation information, traditional systems must make assumptions on how users value computational resources and that the cumulative effect of bad assumptions is significant. On the other hand, compared to coarse-grain, static valuations, we saw that significant benefits are achieved only for parallel workloads. This suggests that supercomputing center policies are effective as long as queues are priced so that queueing delays in high priority queues are small. Finally, we showed that preemption does not improve performance since explicit valuations already decrease queueing delays substantially for important jobs, thereby delivering most of the performance gains.

References

[1] Kento Aida, Hironori Kasahara, and Seinosuke Narita. Job scheduling scheme for pure space sharing among rigid jobs. In *Proceedings of 4th Workshop on Job Scheduling Strategies For Parallel Processing*, March 1998.

[2] Su-Hui Chiang and Mary Vernon. Production job scheduling for parallel shared memory systems. In *Proceedings*

of 15th International Parallel and Distributed Processing Symposium, April 2001.

[3] Brent N. Chun and David E. Culler. Market-based proportional resource sharing for clusters. Technical Report CSD-1092, University of California at Berkeley, January 2000.

[4] Donald Ferguson, Yechiam Yemimi, and Christos Nikolaou. Microeconomic algorithms for load balancing in distributed computer systems. In *International Conference on Distributed Computer Systems*, 1988.

[5] Andrew Geweke. A system for batch-mode economic scheduling of a cluster of workstations. Master's thesis, University of California at Berkeley, 2001.

[6] Mark S. Miller and K. Eric Drexler. *The Ecology of Computation*, chapter 10: Incentive Engineering for Computational Resource Management. Elsevier Science Publishers, October 1988.

[7] Ian Stoica, Hussein Abdel-Wahab, and Alex Pothen. *Lecture Notes in Computer Science, Vol. 949*, chapter A Microeconomic Scheduler for Parallel Computers, pages 200–218. Springer-Verlag, 1995.

[8] Michael Stonebraker, Robert Devine, Marcel Kornacker, Witold Litwin, Avi Pfeffer, Adam Sah, and Carl Staelin. An economic paradigm for query processing and data migration in mariposa. In *3rd International Conference on Parallel and Distributed Information Systems*, pages 58–67, September 1994.

[9] Achim Streit. On job scheduling for hpc-clusters and the dynp scheduler. In *Proceedings of 8th International Conference on High Performance Comtuping*, December 2001.

[10] Carl A. Waldspurger, Tag Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and Scott Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–177, February 1992.