Eliciting Honest Value Information in a Batch-Queue Environment

Andrew Mutz #1, Rich Wolski #2, John Brevik *3

#*Department of Computer Science, University of California Santa Barbara*
Santa Barbara, CA 93106
U.S.A.
1amutz@cs.ucsb.edu
2rich@cs.ucsb.edu

**California State University, Long Beach*
1250 Bellflower Blvd. Long Beach CA 90840
U.S.A.
3jbrevik@csulb.edu

***Abstract*— Markets and auctions have been proposed as mechanisms for efficiently and fairly allocating resources in a number of different computational settings. Economic approaches to resource allocation in batch-controlled systems, however, have proved difficult due to the fact that, unlike reservation systems, every resource allocation decision made by the scheduler affects the turnaround time of all jobs in the queue. Economists refer to this characteristic as an "externality", where a transaction affects more than just the immediate resource consumer and producer. The problem is particularly acute for computational grid systems where organizations wish to engage in service-level agreements but are not at liberty to abandon completely the use of space-sharing and batch scheduling as the local control policies. Grid administrators desire the ability to make these agreements based on anticipated user demand, but eliciting truthful reportage of job importance and priority has proved difficult due to the externalities present when resources are batch controlled.**

In this paper we propose and evaluate the application of the *Expected Externality Mechanism* as an approach to solving this problem that is based on economic principles. In particular, this mechanism provides incentives for users to reveal information honestly about job importance and priority in an environment where batch-scheduler resource allocation decisions introduce "externalities" that affect all users. Our tests indicate that the mechanism meets its theoretical predictions in practice and can be implemented in a computationally tractable manner.

I. Introduction

It has long been recognized that the problem of fairly and efficiently allocating resources on computational grids is one that shares much with traditional problems in the field of economics [1], [2], [3]. A variety of markets and auctions (two broad and overlapping classes of economic mechanisms) have been proposed to determine which jobs will have access to which grid resources, and for how long [4], [5], [6], [7], [8]. The advantage normally associated with using economic mechanisms is that by providing currency-based consequences for resource consumption, users are motivated to avoid inefficient resource usage. Usage that can be discouraged by such mechanisms includes, for example, low-priority jobs being run during periods of high-demand, or processes being run on hardware for which they are not well-suited.

One non-economic mechanism that is commonly in use today in grid environments is the batch queue. In their simplest form, batch queues consist of virtual lines that users wait in for access to computational resources. Batch queues have many strengths: they are simple to use, they produce very egalitarian resource allocations, and they can yield a high degree of resource utilization. They are far from perfect, however, as they provide poor incentives for users to moderate usage during periods of high demand. As a result, high batch queue utilization numbers can belie hidden inefficiency as resources are being allocated to the execution of the job at the front of the queue, regardless of the value of the work to be performed.

In this paper, we present a scheduler that bridges this gap between value-centric economic mechanisms and value-agnostic batch queues. This mechanism is based on the *Expected Externality Mechanism*, an auction mechanism developed independently in 1979 by both Kenneth J. Arrow [9] and by Claude dAspremont and Louis-Andre Gerard-Varet [10]. Due to this theoretical basis, we have named our scheduler the *Expected Externality Scheduler* (or EES).

This paper is organized as follows. In Section II, we describe in greater depth the *external* nature of the participants' preferences in a batch-queued environment. In Section III, we discuss how our work relates to other work in this field. In Section IV, we present the Expected Externality Mechanism and describe how we apply its theoretical form in practice in creating the Expected Externality Scheduler. In Section V, we present the results of our evaluation of the scheduler. We then conclude with a recap of our findings and a description of our intentions for future research in this area in Section VI.

II. Context and Problem

In their simplest form, batch queues are a series of jobs that wait in line for access to resources. The jobs are prioritized by a scheduler policy (set by the owner of the resource) but jobs of equivalent priority are typically handled in a *first-in-first-out* manner by the scheduler, and all jobs commonly have a limitation on their running time. The resource management scheme that we are presenting in this paper maintains uses an alternative approach. All jobs are considered in *first-in-first-out* order and currency-based auction system that allows queued jobs to pay preceeding jobs to not run. By controlling the allocation of currency, the resource owner can control priority.

Building such a system in our target environment is difficult, however, for two primary reasons.

Users are not Altruistic: Computational grids are commonly composed of federated resources from multiple institutions. The users that consume grid resources are themselves from different institutions. In this situation, it is unrealistic to expect users to act in a purely altruistic fashion. If we were to simply ask each user how important their jobs were, without any consequences for overstating the value, it would be in their interest to reply that their jobs were maximally important. Using such responses to make scheduling decisions would, of course, not produce efficient results. In order to be effective, a scheme to manage these resources must assume that users will act in a manner that advances their own best-interests.

Batch Queues have Externalities: In order to select an appropriate economic mechanism for a batch-queued environment, it must be observed that the decision to run the job at the front of the queue affects all users. If the job runs, all other jobs deeper in the queue will be delayed by the running time of that frontmost job. When a transaction affects more than just the resource consumer and producer, economists refer to this effect as an *externality*.

A classic example is the polluting factory. The factory produces goods and sells them to consumers, but the pollution generated by the factory affects more than just those who buy the product: everyone living nearby is affected by the results of transactions in which they do not directly participate. An economic model that does not consider these external effects may not capture the true incentives of the participants or accurately represent the true valuations assigned to resources resources.

In terms of mechanism design, this external nature of the participants' preferences significantly limits the space of mechanisms for trading goods that we have at our disposal. For example, most auction and market mechanisms in use today don't consider preference externalities. As a result, in some industries, where the unconsidered preference externalities are very strong, the government (an altruistic organization, economically speaking) must pass legislation in order to rectify the problems that result.

In a batch scheduling computational environment, self-interested users and the externalities that queuing introduces make it difficult to build a resource management system that elicits honest information from users about job priority. Indeed previous work indicates that users routinely provide inaccurate information even when doing so may harm the priority given to their own workload [11]. Our work explores a new approach to this problem, however, which we present more fully in the next section.

III. Related Work

Previous work has documented the gains in user satisfaction when job value is accounted for in scheduling decisions. In [12], Chun and Culler simulate different scheduling systems that account for job value when making decisions and find that aggregate user satisfaction can be increased by a factor of between 2 and 14 over value-agnostic schedulers. Additionally, in [13], Lee and Snavely show that users are capable of expressing complex preferences when asked. The authors surveyed users at the San Diego Supercomputer Center about expected running times and about the value of different turnaround times and found that while users generally have a poor ability to predict running time, users are able to express in great detail the value they see as a function of turnaround time.

Many scheduling systems have been proposed to account for this job-value information when making decisions. Some scheduling systems rely on pricing resource *reservations*, where users are promised access to future resources. In [14], Bubendorfer *et al.* propose a reservation system that uses the Generalized Vickrey Auction to price resource reservations on computational grids. In [15], Wellman et al discuss the strengths and weaknesses of different auction design choices as they affect pricing generic resource reservation slots, including in their analysis variants of Ascending Auctions and the Generalized Vickrey Auction.

Many scheduling systems for computational clusters eschew reservations and instead distribute resources in a spot market where jobs compete for access to computational resources. As with reservations, the Generalized Vickrey Auction is commonly used: both Schnizler et al in [6] and Das *et al.* in [16] use an approximated version of it for spot market distribution, although they use different approximation techniques. A similar mechanism is pursued by Lai *et al.* in [5], although instead of the Generalized form they use non-combinatorial second-price auctions. The distribution of grid resources by commodities markets are explored for this application by Wolski *et al.* in [8], where both the *Tatonnement* and *Smale* price adjustment mechanisms are compared to auctions. An original auction mechanism is presented by Chun *et al.* in [4], where rather than bidding for complete control of a resource, jobs are allocated to resources in proportion to the quantites of their bids.

Rather than distributing these resources as reservation slots or on a spot market, this paper focuses on an auction scheme that can be used in a batch-queued environment. The application of pricing mechanisms to batch queues has been pursued before. In [12], Chun *et al.* compare traditional, value-agnostic queuing systems to a scheme that reorders jobs in the queue based on bids. In [7], Stoica *et al.* use a first-price auction to determine which job in a *ready list* will run, and then use a combination of those bids and price prediction to determine how much participants pay.

Much work has preceded ours in the application of economic mechanisms to the allocation of grid resources. Our work is novel, however, in the that we treat the queueing externalities in the system as a significant factor that must be directly accounted for in the auction mechanism we are using. Our work is also novel in that we believe it to be the first application of the Expected Externality Mechanism to the allocation of computer resources.

IV. Mechanism Description

As discussed in Section II, our Expected Externality Scheduler maintains the *first-in-first-out* semantics of a non-prioritizing batch queue and relies on the allocation of currency to set priorities. On submission, a job is placed at the back of the queue. The user submitting the job specifies to the resource management system currency-based parameters that indicate the priority of the job. Whenever the resource becomes available, the scheduler considers running the frontmost job and decides, based on that job's specified priority information and the information associated with all jobs deeper in the queue, to either run the job or to discard the job and consider the next in line.

In order for this priority mechanism to make this decision effectively, the priority information supplied by the users must be accurate. To do this, we employ the *Expected Externality Mechanism*, a general auction mechanism developed independently in 1979 by both Kenneth J. Arrow [9] and by Claude dAspremont and Louis-Andre Gerard-Varet [10]. We use this result to design a batch-scheduling queuing protocol in which users express the value of their job in terms of a "bid" when the job is enqueued. Each user is either subsequently charged currency when her job is ultimately run or receives a payment (compensating for externalities) if the bid was insufficient to allow the job to run. These charges and payments are computed in a very specific fashion, so that the following properties hold.

Users see the greatest benefit when they accurately specify the value of their jobs: As discussed previously, it is essential that the mechanism guarantees that it is in the best interests of a participant to honestly reveal her value information. The *Expected Externality Mechanism* appeals to game theory to prove that its use guarantees this property, although the nature of this guarantee takes a specific form. In game-theoretic terms, the strongest form of this guarantee would be if honesty was a *dominant strategy*, meaning honesty was an optimal strategy to employ, regardless of the strategies employed by the other users participating in the system. The Expected Externality Mechanism, however, provides a slightly weaker form of guarantee – one where truth-telling is a *Nash-optimal strategy*, meaning it is an optimal strategy to employ if other participants are also reporting their preferences honestly. While truth revelation as a dominant strategy is clearly desirable, practically we know of no market mechanism at present that is able to achieve it in a setting where externalities exert are as strong an influence as they do in a resource service queue. We discuss how a scheduler can implement an Expected Externality Mechanism while staying at a Nash-optimal equilibrium.

Payments are zero-sum: In order for participants to be able to accurately specify the value the jobs they are submitting, it is important that there not be large inflows or outflows of currency in the system. The balanced-budget property provided by this mechanism avoids this problem by assuring us that the total amount of currency in the system remains constant over time.

Payments are computed using a computationally tractable algorithm : Many auction mechanisms in this space require the solution of NP-Hard problems in order to compute the payments and allocation. Such a requirement is clearly an unacceptable characteristic for a mechanism that is to be deployed in a high-performance computing environment.

A. Job Priority Model

Our scheduler uses two currency-based parameters to indicate job priority: the value of the work to be performed, and the tolerance of the user towards delay in total turnaround time. We will refer to these parameters as v and d, respectively. Both v and d are specified in terms of a cluster-specific currency that has been allocated by the administrators of the resource. In addition to this value information, the user indicates a maximal running time of the job, r.

The value parameter, v, is simply the amount of currency that the user would be willing to pay in order to have the job executed. High-value jobs have a low likelihood of being discarded by the mechanism when they reach the front of the queue.

The delay tolerance parameter, d, represents how eager a user is to have other preceding jobs discarded by the mechanism. This parameter can be thought of as being in units of currency-per-second, where the amount of dissatisfaction that is experienced by that user by waiting x seconds is $x * d$. The presence of many jobs in the queue with high delay tolerance parameters would mean a high likelihood of jobs being discarded by the mechanism.

The decision function that determines whether to run or discard jobs takes all three parameters into account. When considering whether to run job i, that is at the head of the queue it computes:

$$\begin{aligned} a &= v_i && (1) \\ b &= \sum_{j \neq i} d_j * r_i && (2) \end{aligned}$$

$$\kappa^*(a,b) = \begin{cases} discards, & a < b \\ runs, & a \geq b \end{cases} \quad (3)$$

If $a \geq b$, job i is run by the scheduler. Otherwise, job i is discarded and the next job in the queue is considered.

B. Payment Function

After determining whether or not to run the frontmost job, the Expected Externality Mechanism is employed to determine the appropriate payments. The payment mechanism uses information about the statistical distribution of user parameter values in this computation. Following the notation of [17], the payment of each job i is as follows:

$$\left(\frac{1}{I-1}\right)\sum_{j\neq i} E_{s_{-j}}\left[\sum_{l\neq j} u(s_l,\kappa^*(t_j,s_{-j}))\right] \tag{4}$$

$$-E_{s_{-i}}\left[\sum_{j\neq i} u(s_j,\kappa^*(t_i,s_{-i}))\right] \tag{5}$$

where for a job l at the front of the queue,

$$u(s_l,\kappa^*(t_j,s_{-j})) = \begin{cases} v_l, & \kappa^*(t_j,s_{-j}) = runs \\ 0, & \kappa^*(t_j,s_{-j}) = discards \end{cases} \tag{6}$$

and for any job l not at the front of the queue,

$$u(s_l,\kappa^*(t_j,s_{-j})) = \begin{cases} -d_l * r_i, & \kappa^*(t_j,s_{-j}) = runs \\ 0, & \kappa^*(t_j,s_{-j}) = discard \end{cases} \tag{7}$$

In this expression, κ^* is the decision function described in Formula 3. t_m is the parameter stated by job m (in the case of the frontmost job, this is the v, for all other jobs this is d). In this expression, $E_{s_{-j}}(x)$ refers to taking the expected value of x over the possible parameter announcements of all jobs other than j. In this sense, t_m and s_m are variables of similar type, the difference being that t_m is the actual announced parameter of user m, whereas s_m is the parameter corresponding to job m that is drawn during the expected value computation. I is the number of jobs in the queue. The function u quantifies the amount that a user is affected by the allocation decision κ^* in terms of currency.

In our implementation, we use a Monte-Carlo sampling method for numerically evaluating the expected value computation. To do this, we repeatedly draw samples from the parameter distribution associated with each user and use those parameters to evaluate the expression inside the expected value. We then take the arithmetic mean of the result produced by each draw as the expected value. The number of samples necessary to produce an acceptable level of accuracy is a question that is discussed further in Section V-B. Because the expected value computation appears in both terms of the payment function, and because there are no nested expected value expressions, we would expect the running time of the algorithm to scale $O(n)$ linearly as we increase the number of samples we draw when evaluating these expressions. In Section V-B, we show results that confirm this performance feature.

We expect the computation of the payment function to scale $O(n^3)$ cubically as we increase the number of jobs in the queue. This is because the allocation decision κ^* (Formula 3) scales linearly with the number of jobs, and its computation is inside two nested loops whose size also scale linearly with the number of jobs. In Section V-B, we show results that confirm this expectation.

Because the Expected Externality Mechanism requires the knowledge of the statistical distribution of users value information, any system that employs the mechanism would need to know this information. This can be done in practice by the scheduler. The scheduler can exploit the fact that truthful revelation of each job's value is a Nash equilibrium, and retain historical information on each user's preferences. This scheduler can then use this historical data to determine the statistical distribution of participants' preferences. The Expected Externality Mechanism assumes that while users are not aware of other user's announced parameters, users are aware of the distribution of these parameters. As such, it would be appropriate for a deployed system to make this distribution information publicly available to all users in the system.

C. Scheduler Protocol

A batch scheduler implementing this mechanism works as follows. Users are provided with some allocation of currency that represents the cumulative priority of the work they are authorized to run on a machine relative to other users [1]. When a job is submitted, as part of its submission script it must specify its value, its delay tolerance, and its running time. Once submitted, these values cannot be changed and the job cannot be cancelled until it is considered for execution.

Each job waits in the queue as the jobs in front of it are considered for execution. During this time, its d parameter is used in determining whether to run the job that is currently at the front of the queue, as described in Formula 3. Whenever the resource becomes available, the allocation decision function given by Formula 3 is evaluated for the frontmost job. If, as described in Section IV-A, $a \geq b$, the frontmost job runs and the job's owner pays out currency to other users as defined by Equation 4. Otherwise, the frontmost job is cancelled, and no currency changes hands. The user of the discarded job is then free to resubmit (possibly changing the bid for the job). One important observation is that jobs that are cancelled did accrue wealth from the period of time they spent in the queue, as each job that ran during that period paid currency to all queue residents.

In our current implementation, we restrict the number of nodes requested by a job to be the same for every job in the queue. This restriction is due to the fact that for the theoretical guarantees of the Expected Externality Mechanism to hold, the allocation function κ^* must be solved optimally. Allowing jobs of multiple sizes can yield allocation functions that cannot be solved optimally in polynomial time. While this requirement may be overly restrictive for some practical settings, in most workloads, a histogram of node counts typically shows that a large fraction of jobs require a power of 2 number of processors. It may be that by restricting the number of queues servicing a machine to one for each power

[1]This method of prioritization is currently practiced at many computing centers where users apply for allocations of node-hours that they can then "spend" by running jobs over some fixed time period. When a job is executed, the product of its time and the number of nodes it uses are decremented from the allocation account.

of 2 we have achieved a more practically appealing solution that approximates the optimal method in polynomial time. Additionally, we are currently investigating ways to allow jobs of multiple sizes in a single queue in the Expected Externality Scheduler. A first step we plan to take is to maintain the first-in-first-out characteristics of the current scheduler, and to only allow separate jobs concurrent access to resources if they do not require re-ordering of the queue. This change would only introduce modifications to Formula 7, as the effects of running a given job on the turnaround time of other jobs would be more complex.

The Expected Externality Scheduler protocol has two important properties. First, truth revelation is a *Nash equilibrium* which means that any user's optimal strategy is to reveal her true information regarding each job as long as all other users in the system do so as well. To assure this property so that users are continually induced to reveal their true job characteristics, the system must start in a state where all users (perhaps altruistically) reveal their true information so that deviation from this strategy subsequently is not in any user's best interest. We believe this condition can be realized practically by starting it (e.g. after a scheduler resetart, preventive maintenance, etc.) with a set of jobs that are determined ahead of time to report their values accurately. These jobs might belong to specially trusted users or to the administrative body itself which often has work it wishes to run, and also has a n incentive to see the scheduler remain in a Nash equilibrium.

Secondly, the overall system is *budget-balanced*, in other words, the sum of all payments is zero. This property is useful because it assures the center that its intended prioritization among different users can be realized by the initial currency allocation. Currency may change hands temporarily, inverting priorities in the same way that lower-priority jobs must sometimes be allowed run before higher-priority submissions in order to avoid starvation. Over time, however, the currency does not inflate nor deflate, thus preserving its power over prioritization.

Theoretical proofs of the Expected Externality Mechanism's budget balance and truth revealing properties can be found in [17]. In the next section, we demonstrate these properties empirically.

V. Evaluation

In this section, we detail our efforts to verify that the application of the Expected Externality Mechanism to the problem of batch-scheduling produces the properties predicted by its theoretical formulation. To do so, we resort to simulation, not because we have not been able to construct a batch queue simulator that implements the scheduling protocol described in Subsection IV-C, but because it is not possible to investigate these properties empirically without first investigating them (thoroughly) in simulation. At present, most centers responsible for administering large-scale batch-controlled resources are primarily concerned with the maximization of two metrics: resource utilization, and user contentment. As they are not tasked with forwarding computer science research as a primary mandate, they are understanably reluctant to explore new approaches, particularly when those new approaches will impact user experience as directly as ours will. Thus until our Expected Externality Scheduler has been successfully vetted by the knowlegable research community and also has produced a set of compelling simulation results, it is unlikely that a fully empirical test using a "live" machine and user community will be possible. In this section, we attempt to lay the foundation upon which such a test could be justified.

Theory of Expected Externality predicts that the running time of the mechanism will scale as $O(n^3)$ where n is the number of jobs in the queue. This running time is also influenced by how thoroughly we numerically compute the expected value expressions in the mechanism, and theory predicts that the running time should scale $O(n)$ linearly as we increase the number of draws in this computation. In Subsection V-B, we show that our testing verifies both of these predictions.

In addition to the performance predictions, users in the Expected Externality Scheduler should find that honest revelation of the value of their jobs is a *Nash equilibrium*. Empirically demonstrating that this is the case across all possible combinations of user preferences is not feasible, however, all of our randomly selected simulations did show this to be the case. In Subsection V-C, we present one such example verifying the optimality of the truth-telling strategy.

A. Testing Environment

The following experiments were conducted on a quad-processor Intel Xeon running at 3.2 ghz with 1GB of RAM. The operating system was Debian Linux version 1:3.3.5-13. The simulator was written in the Java programming language and was compiled and run with Sun Microsystem's java distribution version 1.4.2-03.

B. Performance

As mentioned above, we expect to see an $O(n^3)$ cubic relationship between the execution time of the Expected Externality Scheduler and the number of jobs currently waiting in the batch queue. Figure 1 depicts the results of our experiments to test this. The solid line connects the data points that we measured as we increased the number of jobs in the queue. The dashed line is an overlaid cubic function ($y = 0.5947 * x^3 + 3424.42$) that tightly fits the data with a Root Mean Square Error of 4549.13. From the figure, as we increase the number of jobs waiting in the queue, the running time of the algorithm scales in a manner consistent with that predicted by theory.

Some readers may notice that, while the results scale well, the absolute amount of execution time required is quite high (to compute the payments with 244 queued jobs, the simulator ran for over two hours). It is important to note that this absolute execution time depends on the granularity of the expected value computations being performed. The tests in Figure 1 were done with very fine-grained expected value computation

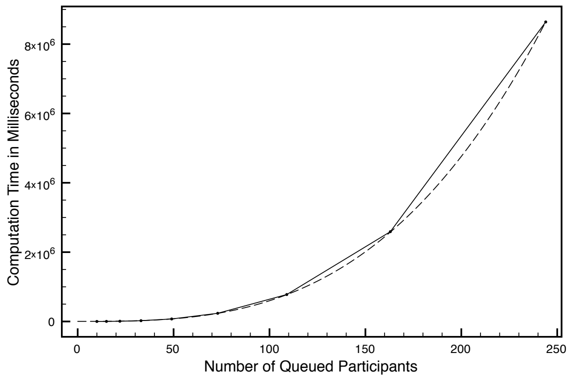

Fig. 1. Demonstration of the $O(n^3)$ running time of the mechanism

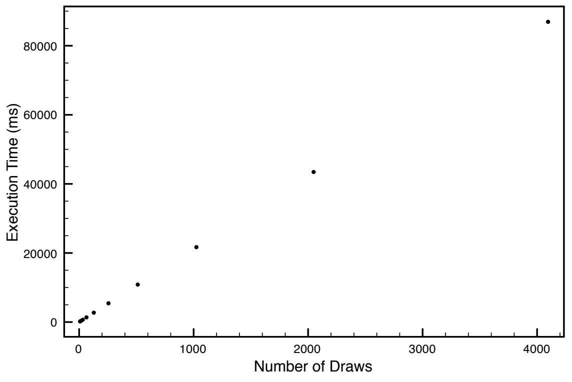

Fig. 2. Execution time with respect to the number of expected value draws

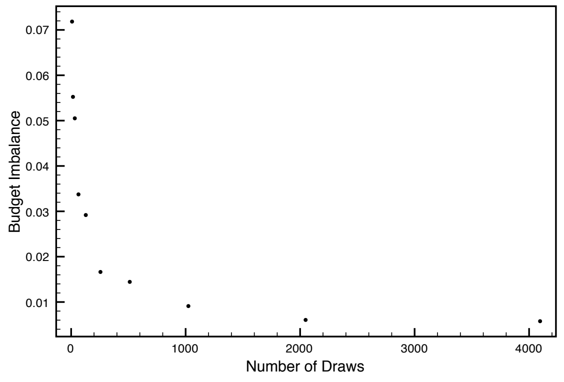

Fig. 3. Budget imbalance with respect to the number of expected value draws

and performed 1000 draws from the associated distribution for each of such computations.

In Figure 2, we demonstrate that this execution time can be reduced by choosing a more coarse-grained level of expected value computation. We tested the amount of time required by the mechanism on a simulated queue of 32 jobs as we scaled the degree of granularity. Each data point in the graph represents the execution time for each granularity level of expected value computation. The figure clearly shows a linear relationship between the number of draws and the resulting execution time.

This reduced granularity does not come without consequences, however. In Figure 3, we show the relationship between the expected value granularity level and the degree to which the payments in the system balance out. Each dot in the graph is the budget discrepancy divided by the total magnitude of payments in the system. As described in Section IV, one of the advantages of the Expected Externality Mechanism is that the sum of the payments in the system is zero. As we reduce the granularity, and as a result the accuracy, of these computations, the consequence is a greater degree of budget imbalance. In this example, we found a budget discrepancy of 7% when we used a very coarse-grained number of 8 draws. The discrepancy continued to drop to below 1% for 1024 draws and appears to asymptotically approach 0% as we increase this granularity further.

We see this performance tradeoff as a strength of the system. The expected value granularity level provides a tunable parameter that allows us to trade payment accuracy for increased performance.

C. Honest Preference Revelation

In the Section IV, we described how the Expected Externality Mechanism provides theoretical guarantees that honest preference revelation is a Nash equilibrium, meaning that it is an optimal strategy if all other participants are being honest. In order to demonstrate empirically that our mechanism has this property, we created simulated queues and explored the expected payoff for many different preference declarations. In each case, our results show that expected payoff was maximized when users honestly declared their preferences.

Figure 4 shows an example of one such run. We created a participant at the front of the queue whose preferences were defined by the parameter 129.8 (meaning the value of the job that was submitted). We then found the expected payoff that the participant would see from randomly selected groups of other participants for each of many declared types. As depicted in the graph, all other non-honest declarations lead to either a comparable or lesser expected payoff for the participant in question. Each small dot in the graph indicates one expected value computation, the large dot indicates the average of the black dots for each declaration.

Figure 5 is a similar graph for a participant further back in the queue. The participant's actual quantification of how much they dislike waiting an additional hour is 48.9, but we explore the payoff seen when declaring many different values. Similar to Figure 1, no other declaration produces a higher expected payoff.

As a result, both of these participants see no value from being dishonest, and in this example honesty is a Nash equilibrium.

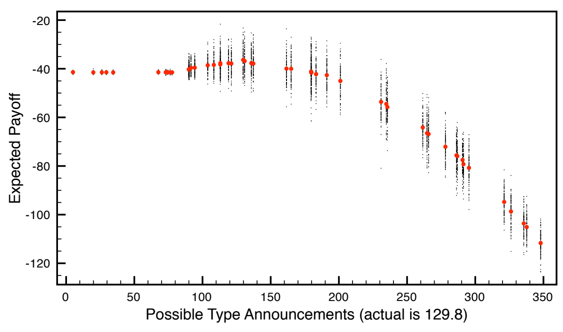

Fig. 4. Expected payoff for the user at the front of the queue

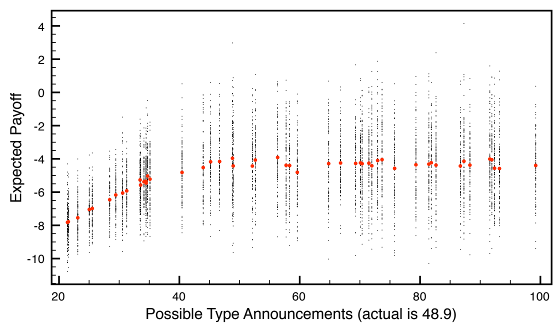

Fig. 5. Expected payoff for a queued participant

VI. Conclusion

In this paper we have documented our efforts to account for the externalities that affect users in batch queued environments. As described in Section II, decisions made about scheduling a job will affect all other users in the batch queue. In order to deploy an economic mechanism to govern batch queues, we need to select one that appropriately accounts for these externalities. In selecting the Expected Externality Mechanism, we are able to build a scheduler that handles these externalities while providing computational tractability, balanced budgets and properly incentivizes users to honestly reveal the value of the work being performed. Additionally, we showed through simulation that the aforementioned properties are not only predicted in theory, but are provided in practice.

We are planning to continue our work in this area. As discussed in Section IV-C, we are investigating how best to allow jobs requesting different numbers of nodes to be handled by the same queue. Because the theoretical guarantees provided by the Expected Externality Mechanism require the allocation function κ^* to be solved optimally, expanding this functionality while retaining the current scheme's computational tractability is not easy. We do have ideas on how to do this, however, and plan to build and evaluate a version of this scheduler with those features.

In addition to improving the feature set of the scheduler, we plan to deploy the scheduler in the real world and observe how it behaves with real users. Game theory predicts that users will honestly reveal the priority information of their jobs, but humans do not always act completely rationally. Only by measuring a live, deployed system can we determine if the scheduler lives up to its theoretical predictions about user behavior.

References

[1] J. F. Kurose and R. Simha, "A microeconomic approach to optimal resource allocation in distributed computer systems," *IEEE Transactions on Computers*, vol. 38, no. 5, pp. 705–717, 1989.

[2] I. E. Sutherland, "A futures market in computer time," *Communications of the ACM*, vol. 11, no. 6, pp. 449–451, June 1968.

[3] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta, "Spawn: A distributed computational economy," *IEEE Trans. on Software Engineering*, vol. 18, no. 2, pp. 103–117, February 1992.

[4] B. N. Chun and D. E. Culler, "Market-based proportional resource sharing for clusters," Computer Science Division, University of California at Berkeley, Tech. Rep. CSD-1092, January 2000. [Online]. Available: http://citeseer.ist.psu.edu/chun99marketbased.html

[5] K. Lai, B. A. Huberman, and L. Fine, "Tycoon: A distributed market-based resource allocation system," 2004.

[6] B. Schnizler, D. Neumann, D. Veit, and C. Weinhardt, "A multiattribute combinatorial exchange for trading grid resources," in *Proceedings of the Research Symposium on Emerging Electronic*, 2005.

[7] I. Stoica, H. Abdel-Wahab, and A. Pothen, "A microeconomic scheduler for parallel computers," in *Proceedings of the International Parallel Processing Symposium (IPPS) '95 Workshop on Job Scheduling Strategies for Parallel Processing*, Santa Barbara, CA, USA, 1994. [Online]. Available: http://citeseer.ist.psu.edu/stoica94microeconomic.html

[8] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan, "Analyzing market-based resource allocation strategies for the computational grid," *International Journal of High Performance Computing Applications*, vol. 15, pp. 258–281. [Online]. Available: http://citeseer.ist.psu.edu/wolski00analyzing.html

[9] K. J. Arrow, *Economics and Human Welfare*. Academic Press, 1979, pp. 23–39.

[10] C. d'Aspremont and L.-A. Gerard-Varet, *Aggregation and Revelation of Preferences*. North Holland Publishing Company, 1979, ch. On Bayesian Incentive Compatible Mechanisms, pp. 269–288.

[11] C. Lee, Y. Schwartzman, J. Hardy, and A. Snavely, "Are user runtime estimates inherently inaccurate," in *Proceedings of Workshop on Job Scheduling Strategies for Parallel Processing*, June 1994.

[12] B. N. Chun and D. 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*, Berlin, Germany, May 2002. [Online]. Available: http://berkeley.intel-research.net/bnc/papers/ccgrid02.pdf

[13] C. B. Lee and A. Snavely, "On the user-scheduler dialogue: Studies of user-provided runtime estimates and utility functions," vol. 20, pp. 495–506.

[14] K. Bubendorfer, K. Chard, P. Komisarczuk, and A. Desai, "Fine grained resource reservation and management in grid economies," in *Proceedings of The 2005 International Conference on Grid Computing and Applications*, 2005.

[15] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason, "Auction protocols for decentralized scheduling," *Games and Economic Behavior*, vol. 35, no. 1-2, pp. 271–303, 2001. [Online]. Available: http://citeseer.ist.psu.edu/383290.html

[16] A. Das and D. Grosu, "Combinatorial auction-based protocols for resource allocation in grids," in *Proceedings. 19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[17] V. Krishna and M. Perry, "Efficient mechanism design," 1998, working paper. [Online]. Available: http://ratio.huji.ac.il/dp/dp133.pdf