High Performance Resource Allocation Strategies for Computational Economies

Kyle Chard, Member, IEEE, and Kris Bubendorfer, Member, IEEE

Abstract—Utility computing models have long been the focus of academic research, and with the recent success of commercial cloud providers, computation and storage is finally being realized as the fifth utility. Computational economies are often proposed as an efficient means of resource allocation, however adoption has been limited due to a lack of performance and high overheads. In this paper, we address the performance limitations of existing economic allocation models by defining strategies to reduce the failure and reallocation rate, increase occupancy and thereby increase the obtainable utilization of the system. The high-performance resource utilization strategies presented can be used by market participants without requiring dramatic changes to the allocation protocol. The strategies considered include overbooking, advanced reservation, just-in-time bidding, and using substitute providers for service delivery. The proposed strategies have been implemented in a distributed metascheduler and evaluated with respect to Grid and cloud deployments. Several diverse synthetic workloads have been used to quantify both the performance benefits and economic implications of these strategies.

Index Terms—Economic resource allocation, utility computing, cloud computing, Grid computing

1 INTRODUCTION

Cloud computing has managed to achieve what was once considered by many to be a pipe dream, that is, it has successfully applied economic (utility) models to commodify computation. Consumers can now outsource computation, storage and other tasks to third party cloud providers and pay only for the resources used. At present the models employed are simplistic (generally posted price), however there are moves toward more sophisticated mechanisms, such as spot-pricing, and a global computation market is a future possibility. A global computation market could be realized by a high-performance federated architecture that spans both Grid and cloud computing providers, this type of architecture necessitates the use of economic aware allocation mechanisms driven by the underlying allocation requirements of cloud providers.

Computational economies have long been touted as a means of allocating resources in both centralized and decentralized computing systems [1]. Proponents of computational economies generally cite allocation efficiency, scalability, clear incentives, and well-understood mechanisms as advantages. However, adoption of economies in production systems has been limited due to criticisms relating to, among other things, poor performance, high latency, and high overheads.

Overheads are, in many ways, inherent in computational economies, for example, in a competitive economy resources are typically “reserved” by m participants for the duration of a negotiation. In most cases, there are only n “winning” participants, therefore all other m - n reservations are essentially wasted for the duration of that negotiation. Moreover, there is an opportunity cost to reserving resources during a negotiation, as they will not be available for other negotiations that begin during the interval of the first negotiation. This type of scenario is clearly evident in auction or tender markets, however it can also be seen in any negotiation in which parties are competing against one another for the goods on offer. In any case, this wasteful negotiation process is expensive in both time and cost and therefore reduces the overall utilization of the system.

In this paper, we suggest the application of two general principles to largely address these inefficiencies: first, avoid commitment of resources, and second, avoid repeating negotiation and allocation processes. We have distilled these principles into five high-performance resource utilization strategies, namely: overbooking, advanced reservation, just-in-time (JIT) bidding, progressive contracts, and using substitute providers to compensate for encouraging oversubscription. These strategies can be employed either through allocation protocols and/or by participants, to increase resource occupancy and therefore optimize overall utilization.

Each of the strategies is examined experimentally within the context of a market-based cloud or Grid using the DRIVE metascheduler [2]. Each strategy has been implemented in DRIVE and is analyzed using synthetic workloads obtained by sampling production Grid traces. The analysis presented in this paper focuses on two distinct areas: first the strategies are evaluated with respect to occupancy, allocation, and system utilization to compare their value in a range of economic and noneconomic scenarios. Second, due to the economic motivation of this work, the strategies are evaluated with respect to system and provider revenue to
determine the effect of economic policies on the strategies
techniques themselves and also the effect of these strategies on revenue.

This paper is an extended version of our previous work [3]. This paper extends the analysis of the proposed
strategies by analyzing system performance under a broader range of conditions and examining the economic implications of their use.

2 RELATED WORK

The earliest published computational market was the 1968
futures market [1] that enabled users to bid for compute time
on a shared departmental machine. Over time these market-
based architectures have grown from distributed computa-
tional economies, such as Spawn [4] and Enterprise [5], to
modern brokers, metaschedulers and distributed architectures such as Nimrod/G [6], DRIVE and SORMA [7].

Looking forward, there is great research interest in the
creation of federated computing platforms encapsulating different computation providers. DRIVE, the system used for the experimental work in this paper, is one example of a
such a federated metascheduler and is designed around the
idea of “infrastructure free” secure cooperative markets. Another prominent example is InterCloud [8] which features a generic market model to match requests with providers
using different negotiation protocols (including auctions), in
which context, our strategies could largely be applied. Another alternative approach is spot pricing [9], while this
approach is in some ways similar to an auction (users set a
max price), the fundamentals of operation are sufficiently
different that a different set of strategies would be needed.

Overbooking has been previously used in computational domains as a way to increase utilization and profit [10],
[11]. In [10] overbooking is used to some extent to compensate for “no shows” and poorly estimated task
duration. In [11], backfilling is combined with overbooking
to increase provider profit, where overbooking decisions are based on SLA risk assessment generated from job execution time distributions.

Globus Architecture for Reservation and Allocation
(GARA) [12] was one of the first projects to define a basic
advanced reservation architecture to support QoS reservations over heterogeneous resources. Since this time many
other schedulers have evolved to support advanced reservations, such as Catalina [13] and Sun Grid Engine
[14]. Reservation aware schedulers have been shown to improve system utilization due to the additional flexibility
specified by some consumers, additionally these architecture have realized different reservation aware scheduling
algorithms [15].

Various papers have looked at last minute bidding and
“sniping” in the context of open outcry online auctions [16],
[17]. Typical motivation for last minute bidding is to combat
“shill bidders” (fake bidders raising the price) and incremen-
tal bidding (bidding in increments rather than bidding
ones true value or proxy bidding). JIT bidding for sealed
bid auctions was first proposed in our earlier work [18] as a
means of reducing the effect of auction latency in
distributed auctions, although at the time no experimental
work was undertaken.

The first use of second chance substitutions in a two phase
contract structure was also presented in [18] although in this
work the intention was to reduce coallocative failures. In our
present work, we have adapted and generalized this
mechanism to directly support overbooking to increase
resource utilization.

The general focus of prior work has been on economic
efficiency. In particular, existing systems using auctions
suffer from significant latency and consequently reduced
utilization. In addition, techniques such as advanced
reservation, have in general been approached from a
consumer’s perspective rather than concentrating on the
flexibility given to providers, an exception to this is our
previous work with Netto et al. in [15].

This paper, along with the original conference paper [3],
presents the first coordinated set of strategies to improve
the utilization of resources and negotiation latency when
using auctions for allocation. In addition, this paper further
extends this work by examining the implications of these
strategies on the underlying economic system.

3 OPPORTUNITIES AND HIGH UTILIZATION
STRATEGIES

The life cycle of economic negotiation presents a number of
opportunities to implement utilization improving policies
and strategies before, during, and after negotiation. In a
traditional auction, providers auction resources by soliciting
consumer’s bids, at the conclusion of the auction an
agreement is established to provide resources for the
winning price, when the agreement expires the resources
are returned. Reverse auctions switch the roles of provider
and consumer, therefore mapping more accurately to user
requested resource allocation (e.g., in a cloud). The life cycle
of a reverse auction in DRIVE is shown in Fig. 1. In a reverse
auction a consumer “auctions” a task (or job), providers
then bid for the right to provide the resources required to
host the task. The following high-performance strategies are
defined according to a reverse auction model, however they
could also be applied in a traditional auction model.

3.1 Preauction

3.1.1 Overbooking

There is potential for considerable latency in auction-based
resource allocation, from bidding to resource redemption,
this latency greatly impacts utilization if resources are reserved while waiting for the result of the auction. For example, an auction generally has a single winner, and multiple \( m \) losers. While the winner gains the eventual contract, there is no such compensation for the \( m \) losers of the auction process, and any resources \( r \) put aside during the auction will decrease the net utilization of the system by \( mr \).

From a providers perspective utilization and profit can be increased by participating in auctions that could exceed capacity in the knowledge that it is unlikely they will win all auctions on which they bid. Knowledge of existing bids can also be used in subsequent valuations, thus incorporating the possibility of incurring penalties for breaking agreements.

Overbooking has been shown to provide substantial utilization and profit advantages [10] due to “no shows” (consumers not using the requested resources) and overestimated task duration. While overbooking may seem risky it is a common technique used in yield management and can be seen in many commercial domains, most notably air travel [19], [20] and bandwidth reservation [21]. Airlines routinely overbook aircraft in an attempt to maximize occupancy (and therefore revenue) by ensuring they have the maximum number of passengers on a flight. Without overbooking full flights often depart with up to 15 percent of seats empty [19]. Overbooking policies are carefully created and are generally based on historical data. Airlines acknowledge the possibility of an unusually large proportion of customers showing up, and include clauses in their contracts (agreements) to “bump” passengers to later flights and compensate passengers financially. Due to the widespread adoption of overbooking techniques in commercial domains there is substantial economic theory underpinning appropriate strategy [22], [23].

3.2 During Auction

3.2.1 Just-in-Time Bidding

During negotiation it is possible that resource state may change and therefore invalidate a providers valuation (or bid). In general, there are two ways to minimize the effect of latency:

- Reducing the duration of the auction. The problem with this approach is that there is minimal time for providers to discover the auction and to compute their bids.
- Bid as late as possible. The advantage with this approach is that providers can compute their bids with the most up to date resource state and resources are reserved for a shorter time. The primary problem with this approach is time sensitivity, the auction can be missed if the bid is too late or experiences unexpected network delays.

In some environments, for example, the open outcry protocols used in online auctions, JIT bidding is common and has additional strategic advantages for combating shirking bidding and incremental bidding. For sealed bid auctions, JIT bidding traditionally has been seen to have no advantages. However, this paper highlights the significant utilization improvements that can be obtained by using JIT bidding in a sealed bid resource auction.

3.2.2 Flexible Advanced Reservations

Advanced reservation support is commonly used in distributed systems to provide performance predictability, meet resource requirements, and provide quality of service (QoS) guarantees [12], [24]. As Grid and cloud systems evolve the task of planning job requirements is becoming more complex, requiring fine grained coordination of interdependent jobs in order to achieve larger goals. Often tasks require particular resources to be available at certain times in order to run efficiently. For example, a task may require temporary data storage while executing and more permanent storage after completion. Tasks may also require coordinated execution due to dependencies between one another. In addition to consumer advantages, providers also benefit by being given flexibility in terms of task execution and therefore they have the opportunity to use advanced scheduling techniques to optimize resource usage. We believe this increased flexibility can lead to substantial performance improvements for providers.

3.3 Postauction

3.3.1 Two Phase Contracts

Auction latency may restrict providers participating in future negotiations due to a lack of knowledge of the outcome of ongoing or previous negotiations. There are two general approaches to mitigate this issue, that is, providers can reserve resources for the duration of the negotiation immediately, or they can wait for the result of the allocation before reservation. Neither situation is ideal—initial reservation leads to underutilization as a negotiation typically has one winner and multiple losers, while late reservation results in contract violations as resource state may change between negotiation and reservation. To minimize the effect of latency we propose a progressive two phase contract mechanism that reflects the various stages of negotiation.

The two phase contract structure is shown in Fig. 1. As the result of an allocation a tentative agreement is created between the user and winning provider(s) (phase 1), before redemption this agreement must be hardened into a binding agreement (or contract) that defines particular levels of service to be delivered along with a set of rewards and penalties for honoring or breaking the agreement (phase 2). Typically, the tentative agreement is not strictly binding and penalties for violations are not as harsh as for breaking a binding contract. The motivation for this separation is twofold, first it encourages providers to participate in allocation in the knowledge they will not be penalized as harshly for breaking the agreement at an earlier stage. Second, it facilitates another layer of overbooking (at the time of reservation) as described in Section 3.1.1.

3.3.2 Second Chance Substitute Providers

If a winning provider cannot meet their obligations at the conclusion of an auction (due to overbooking), it is a waste resources to reexecute the auction process when there is sufficient capacity available from nonwinning providers. In this case, the losing bidders can be given a second chance to win the auction, by recomputing the auction without the defaulting bidder. This technique can reduce the allocation failures generated from overbooking and therefore increase
utilization of the system. One negative aspect of this approach is the potential for increased consumer cost, as the substitute price (SP) is, by definition, greater than the previous winning price. However, this cost can be offset through compensation provided by the violating party. Due to the potential impact on auction protocols, all participants must be aware of these semantics prior to the auction.

4 DRIVE

Distributed Resource Infrastructure for a Virtual Economy (DRIVE) [2], [25] is a distributed economic metascheduler designed to allocate workload in distributed and federated computing environments. Allocation in DRIVE is abstracted through an economic market which allows any economic protocol to be used. DRIVE features a novel “co-op” architecture, in which core metascheduling services are hosted on participating resource providers as a condition of joining the Virtual Organization (VO). This architecture minimizes the need for dedicated infrastructure and distributes management functionality across participants. The co-op architecture is possible due to the deployment of secure economic protocols which provide security guarantees in untrusted environments [26].

In DRIVE, each resource provider is represented by a DRIVE Agent that implements standard functionality including; reservations, policies, valuation, and plug-ins for the chosen economic protocol (e.g., bidding). DRIVE Agents use policies and pricing functions to price goods. The DRIVE marketplace includes a number of independent services and mechanisms that provide common functionality including resource discovery, allocation, security, VO management, and contract (agreement) management. DRIVE is designed to support flexible deployment scenarios; it is therefore independent from a particular type of task (e.g., service request, cloud VM or Grid job) in each phase of the task life cycle (submission, allocation, and execution).

DRIVE implements the two phase contract model described in Section 3.3.1. Each agreement defines rewards (and penalties) for honoring (or breaching) the agreement. Each provider also exposes a Reservation Service to plan resource usage and track commitments made by the provider. Reservation information is specified in advance through the initial task description so that providers can take reservation windows into consideration during the allocation phase. Further detail about DRIVE is presented in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.102.

5 EXPERIMENTAL ECONOMY

The pricing functions we use in our experimental economy, as given in Section 5.1, are primarily based upon local information known by the provider and aim to incorporate the perceived risk of a transaction. For example, the price may be increased if a provider has little spare capacity. The penalties, as given in Section 5.2, provide a formal way of categorizing agreement breaches and punishing responsible parties.

5.1 Pricing Functions

All bid prices are determined based on a combination of current conditions, projected conditions or previous bidder experience. In the following equations, \( P_{\text{unit}} \) is the price per unit, \( b \) denotes a bid in the specified range \((0, 3)\). Job units (JU) are defined as the product of CPU utilization and duration \((J_{\text{units}} = J_{\text{utilization}} \times J_{\text{duration}})\). The Random and Constant pricing functions are baseline functions. The other pricing functions attempt to simulate some aspect of a dynamic supply and demand model, in these functions prices are adjusted dynamically to reflect the projected capacity or to factor in previous successes. More specifically:

- Random: the unit price is determined irrespective of any other factors
  \[ P_{\text{unit}} = \text{Random}(0, B) \]

- Constant: the unit price is the same for every request
  \[ P_{\text{unit}} = c, c \in (0, B) \]

- Available capacity: the unit price is calculated based on projected provider capacity at the time when the job would execute. \( U_{\text{provider}} \) is the projected utilization of the provider, \( U_{\text{job}} \) is the utilization of the requested job, and \( C_{\text{provider}} \) is the total capacity of the provider
  \[ P_{\text{unit}} = \frac{U_{\text{provider}} + U_{\text{job}}}{C_{\text{provider}}} \times B \]

- Win/loss ratio: the unit price is based on the previous win/loss ratio seen by the individual provider. \( R \) is the specified ratio, \( W \) is the number of wins since time \( t_0 \), and \( L \) is the number of losses since time \( t_0 \)
  \[ P_{\text{unit}} = (R - L) \times \frac{B}{R} + \frac{B}{2} \]

- Time based: the unit price is based on the time since the provider last won an auction. The unit price decrements every \( T_{\text{period}} \) seconds, \( t_{\text{lastwin}} \) is the time since the last allocation. \( t_{\text{lastwin}} \) is set to 0 at time \( t_0 \)
  \[ P_{\text{unit}} = B - \frac{t_{\text{lastwin}}}{T_{\text{period}}} \]

5.2 Penalty Functions

We classify the following penalty functions into two distinct penalty types: Constant penalties are fixed penalties that are statically defined irrespective of any other factors, whereas Dynamic penalties are based on a nonstatic variable designed to reflect the value of a violation. Dynamic penalties are further classified to model the impact of a violation: \( \alpha \) penalties are based on the relative size of the job or the established price, whereas \( \beta \) penalties attempt to model the increased cost incurred by the consumer using a ratio of the original and substitute prices. \( \beta \) penalties are only possible when second chance substitute providers are used. Specifically, the different penalty functions are:

\[ P_{\text{penalty}} = \begin{cases} \alpha \cdot J_{\text{duration}} & \text{if} \quad P_{\text{unit}} < \text{established price} \\ \beta \cdot (P_{\text{unit}} - \text{established price}) & \text{if} \quad P_{\text{unit}} > \text{established price} \end{cases} \]
- Constant: a constant penalty defined statically irrespective of the job requirements or bid price
  \[ P_{\text{defaulter}} = c, c \in \mathbb{R}_{\geq 0}. \]

- Job units: an \( \alpha \) penalty based on the requirements of the job in units. \( J_{\text{units}} \) is the number of units in a job, and \( c \) is a constant penalty per unit
  \[ P_{\text{defaulter}} = J_{\text{units}} \times c, c \in \mathbb{R}_{\geq 0}. \]

- Win price (WP): an \( \alpha \) penalty based on the winning bid (pre-substitutes). \( \text{Price}_{\text{win}} \) is the price to be paid by the winning bidder:
  \[ P_{\text{defaulter}} = \text{Price}_{\text{win}}. \]

- Substitute price: an \( \alpha \) penalty based on the substitute bid. \( \text{Price}_{\text{substitute}} \) is the price to be paid by the substitute winning bidder:
  \[ P_{\text{defaulter}} = \text{Price}_{\text{substitute}}. \]

- Bid difference (BD): a \( \beta \) penalty defined as the difference between the original win price and the substitute price
  \[ P_{\text{defaulter}} = \text{Price}_{\text{substitute}} - \text{Price}_{\text{win}}. \]

- Bid difference/depth: a \( \beta \) penalty that determines the impact of an individual provider defaulting on a contract. The impact is calculated as the difference between original win price and substitute price evaluated over all defaulters. In the first configuration only a single penalty is applied to the original winning provider, the second configuration imposes a fraction of the penalty on each defaulting provider. \( \text{Depth}_{\text{substitute}} \) is the number of substitutes considered
  \[ P_{\text{defaulter}} = \frac{\text{Price}_{\text{substitute}} - \text{Price}_{\text{win}}}{\text{Depth}_{\text{substitute}}}. \]

  \[ \forall i \in D : \text{Piece}_{i} = \frac{\text{Price}_{\text{substitute}} - \text{Price}_{\text{win}}}{\text{Depth}_{\text{substitute}}}. \]

In general, there is a tradeoff between fairness and complexity of penalty functions. For example, while a constant penalty is easy to enforce and requires no computation it is not fair in terms of which defaulters pay the penalty, it also does not reflect the size or value of the job (both small and large jobs are penalized equally). Application of penalties to each defaulting party is arguably fairer, however it is much more complicated to determine each defaulters effect and to also apply the penalty to multiple parties.

### 6 Evaluation

In this section, each of the strategies is evaluated empirically with respect to allocation occupancy, utilization, and economic implications. We define occupancy as the number of contracts satisfied (i.e., tasks allocated and completed) and utilization as the amount of a host’s resource capacity that is used (in the following experiments we consider CPU as the only measure of capacity). Each strategy is evaluated under different workload scenarios on a testbed deployment.

#### 6.1 Synthetic Workloads

To evaluate the strategies defined in Section 3 we developed several synthetic workloads that simulate different workload conditions. Each synthetic workload is derived from a production Grid trace obtained from AuverGrid, a small sized Grid in the Enabling Grids for E-science in Europe (EGEE) project which uses Large hadron collider Computing Grid (LCG) middleware. It has 475 computational nodes organized into five clusters (each has 112, 84, 186, 38, 55 nodes). This trace was chosen as it was the most complete trace available in the Grid Workloads Archive [27]. While AuverGrid is a relatively small scale Grid the model obtained from the workload can be scaled up to be used in the analysis of these strategies.

The AuverGrid workload is characteristic of a traditional batch workload model, in which jobs arrive infrequently and are on average long running. Using the entire workload as a basis for the following experiments is infeasible due to the duration (1 year) and cumulative utilization (475 processors). There are two ways to use this data: 1) a sample can be taken over a fixed period of time to simulate a workload characterized by long duration batch processing (Section 6.1.1), 2) a synthetic high-performance workload can be generated to reflect a modern fine grained dynamic workload by increasing the throughput while maintaining the general workload model (Section 6.1.2). The dynamic usage model is designed to more accurately represent modern (interactive) usage of distributed systems as seen in Software-as-a-Service (SaaS) requests, workflows, and smaller scale ad hoc personal use on commercial clouds. These two interpretations of the data have been used to generate workloads at either end of the perceived use case spectrum. A summary of the different synthetic workload characteristics is presented in Table 1.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Average Arrival Rate (jobs/hour)</th>
<th>Maximum Arrival Rate (jobs/hour)</th>
<th>Average Run Time (ms)</th>
<th>Average CPU (%)</th>
<th>Job Total Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td>1013</td>
<td>211</td>
<td>527</td>
<td>5152</td>
<td>82.40</td>
</tr>
<tr>
<td>Low Interactive</td>
<td>211</td>
<td>234</td>
<td>357</td>
<td>5818</td>
<td>93.02</td>
</tr>
<tr>
<td>Medium Interactive</td>
<td>4677</td>
<td>519</td>
<td>814</td>
<td>59128</td>
<td>93.67</td>
</tr>
<tr>
<td>High Interactive</td>
<td>4677</td>
<td>1223</td>
<td>1892</td>
<td>59579</td>
<td>93.62</td>
</tr>
</tbody>
</table>

#### Table 1: Experiment Workload Characteristics
The batch workload is generated from a two day sample of the complete AuverGrid trace. The two days chosen include the busiest day (number of jobs) in the trace. This model represents favorable auction conditions as the ratio of auction latency to interarrival time is large. Job duration is on average over 1 hour per job, and jobs use approximately 80 percent of a single processor. Due to the nature of the trace this workload represents large system requirements, the two day workload peaks at 32,709 percent cumulative utilization which is equivalent to 327 nodes completely utilized. Reducing the sample size such that this workload can be hosted on our 20 node testbed is impossible as the resulting number of jobs would be minimal (approximately 600 jobs over 48 hours). Instead, experiments using the batch workload utilize an increased testbed capacity by simulating “larger” providers. In these experiments, a single provider node has the equivalent of 15 processors, and the entire testbed has the equivalent of 300 processors. Maximum individual provider capacity is therefore 1,500 percent. The batch workload and the capacity of the testbed is shown in Fig. 2.

6.1.2 Dynamic Model

Due to the mainstream adoption of cloud computing, usage is evolving from a traditional batch model to a more dynamic on demand model. Modern usage is therefore characterized by extensible, short duration, ad hoc, and interactive usage. This type of usage presents a potentially worst case scenario for auction performance. To simulate this type of high-performance dynamic model we have reduced the time-based attributes of the workload by a factor of 1,000 (i.e., 1 second of real time is 1 millisecond of simulated time). By reducing each parameter equally relativity between parameters is maintained and therefore the distribution is not effected. The performance analysis in this paper looks specifically at the allocation performance without considering the duration of the jobs. However, this reduction in time effects the ratio of interarrival time to auction latency which increases the effect of auction latency.

Three synthetic dynamic workloads have been created to evaluate the proposed strategies under varying degrees of load, the three workloads are summarized in Fig. 3 and Table 1. The resulting workloads, low utilization, medium utilization, and high utilization contain 2111, 4677, and 11,014 jobs, respectively, with each spread over almost 9 hours. The average job runtime for each model is 11,014 jobs, respectively, with each spread over almost 9 hours. The average job runtime for each model is 11,014 jobs, respectively, with each spread over almost 86.55 percent overall utilization which can be completely hosted in the testbed. The medium model slightly exceeds the capacity of the testbed with various points above the available capacity. The high-utilization model greatly exceeds the available capacity of the testbed with most values well above 100 percent. The arrival rate of tasks also increases with each workload as the number of jobs increases over the same period of time. The maximum arrival rate of the medium workload is similar to the maximum arrival rate seen in the original trace, while the arrival rate for the high-utilization workload greatly exceeds the arrival rate of the original trace, with a maximum arrival rate more than double the peak arrival rate seen in the original trace. Each trace typifies the important characteristics of the original trace by providing similar or greater maximum throughput while maintaining the duration of jobs relative to arrival time and one another.

6.2 Experimental Testbed

In these experiments, the testbed is configured with 20 virtualized providers distributed over a 10 machine Grid (five Windows Vista, five Fedora Core) connected by a gigabit Ethernet network. The machines each have Core 2 Duo 3.0 GHz processors with 4 GB of RAM. A single Auction Manager and Contract Manager are run on one host, with each allocated 1 GB of memory. The 20 providers each have 512 MB of memory allocated to the hosting container. Using the dynamic workloads each provider is representative of a single node (100 percent capacity). To satisfy the increased requirements of the batch model providers are configured to represent 15 nodes (1,500 percent) in the batch experiments.

6.3 Strategy Evaluation

The strategies presented in this section are evaluated with respect to the number of auctions completed, contracts created and overall system utilization. The strategies are denoted: Overbidding (O), Second chance substitutes (S), and flexible advanced Reservations (R). In addition a Guaranteed (G) strategy is also implemented against which we compare the other strategies. In the following experiments, a sealed bid second price (Vickrey) protocol is used to allocate tasks, each provider implements a random bidding policy irrespective of job requirements or current capacity. Contracts are accepted only if there is sufficient capacity regardless of what was bid. In the following results, we run each experiment three times and state the average result.

The different strategy combinations examined are designed to isolate particular properties of the strategies and to satisfy dependencies between strategies (e.g., second chance...
providers are only valuable when providers overbid). The major difference between these strategy combinations is related to the options available when calculating a bid, and what actions can be taken at the auction and contract stages of negotiation.

- **G**: Providers bid based on expected utilization, that is they never bid beyond their allotted capacity. As bids are a guarantee, providers cannot reject a resulting contract and therefore there are no opportunities to use second chance substitute providers. This combination does not support advanced reservation, therefore tasks must be started immediately following contract creation.

- **O**: Providers bid based on their actual utilization (irrespective of any outstanding bids), as providers can bid beyond capacity they may choose to accept or reject contracts depending on capacity at the time of contract creation. Second chance substitute providers and advanced reservations are not available in this configuration.

- **S + O**: Providers bid based on their actual utilization, in addition to accepting and rejecting contracts, losing providers may be substituted with a second chance provider at the contract stage if the winning provider does not have sufficient capacity.

- **R + O**: Providers bid based on projected utilization at the time of job execution. This combination allows providers to schedule (and reschedule) tasks according to the defined reservation window, likewise contracts can be accepted if there is sufficient projected capacity during the reservation window defined by the task. No second chance substitutes are considered in this combination.

In each experiment tasks from the workload trace are submitted to DRIVE for allocation. For each task DRIVE conducts a Vickrey auction allowing each provider to bid. At the conclusion of the auction DRIVE determines the winner and attempts to create a contract with the winning provider. Throughout the process the Auction Manager logs information about each auction (including bids and winners), the Contract Manager logs information about contract creation (including rejections and substitute providers), and DRIVE Agents log information about their bids and their current and projected utilization. This information is used to produce the results discussed in this section. Task activity is simulated by each provider based on the utilization defined in the workload trace. Total system utilization is calculated by adding together the (self-reported) utilization for each provider in the system.

Figs. 4, 5, 6, and 7 show an Exponential Moving Average (EMA) of total system utilization for each of the experiments on the dynamic (low, medium, and high) and batch workloads, respectively. These figures are used to compare the strategies against one another throughout this section. Individual allocation results and system utilization for each strategy and workload is also presented in Section 6.3.6.

### 6.3.1 Guaranteed Bidding Strategy

In the baseline configuration, providers implement a guaranteed bidding strategy where every bid by a provider is a guarantee that there will be sufficient capacity. It is
evident in each of the figures that the average and maximum utilization is extremely low. In all workloads, the rejection rate is substantial with only 34.41 percent (low), 26.75 percent (med), 16.96 percent (high), and 47.69 percent (batch) of tasks allocated. As expected these rejections occur during auctioning and no contracts are rejected, as no provider should ever bid outside its means. These results highlight the issue with bidding only on auctions a provider can guarantee to satisfy. A large number of tasks are rejected even though there is sufficient available overall capacity. The reason for this is the number of concurrent auctions taking place and the latency between submitting a bid and the auction concluding. In this testbed, using the dynamic workloads, there is a 0.05 of actually winning. If the auction latency is reduced or the frequency of tasks being submitted is reduced, the number of tasks allocated and total utilization would improve as bidders would have a clearer picture of utilization when bidding on subsequent auctions.

6.3.2 Overbooking Strategy
In the high dynamic workload and the batch model, the peak system utilization approaches the maximum available capacity of the testbed when providers can bid beyond capacity. The average utilization and percentage of tasks allocated for all workloads is more than double that of the guaranteed strategy which highlights the value of overbooking. The allocation improvement exhibited in the batch workload represents the single biggest gain of any strategy and results in near optimal allocation and utilization. In each workload, very few auctions fail (e.g., only 42/11,014 in the high workload) as providers only reach close to maximum capacity for a short period of time. However, the main issue with overbooking is the number of auctions completed that are then unable to be converted into contracts, this approaches 60 percent of all tasks in the high workload, 33 percent in the medium workload, 15 percent in the low workload, and over 5 percent in the batch workload. The number of contracts unable to be established directly effects system performance as the auction and contract negotiation processes are wasted.

6.3.3 Second Chance Substitutes and Overbooking
In the batch model, the use of substitutes reduces the overbooking contract rejection rate by 80 percent and increases overall allocation by 3.71 percent. In the dynamic workloads, average utilization is improved from the overbooking strategy by up to 26 percent (low—17 percent, med—26 percent, high—20 percent) and overall task allocation is increased to 99.94, 85.27, and 51.17 percent for the low, medium, and high workloads, respectively. These results show a large improvement from the previous strategies. Interestingly the depth of substitute providers needed to be consulted is less than 3 on average for each workload indicating that there is ample reserve capacity in the system. This shows that the computational overhead required to fulfill substitution is low.

6.3.4 Reservations and Overbooking
In the AuverGrid workload trace, there is no explicit execution windows, so in order to evaluate reservation strategies and analyze the effect on allocation and utilization we define an execution window for each task as 50 percent of the task duration (sensitivity analysis is presented in [3]). In this experiment providers implement a simple First Come First Served scheduling policy. Each provider again uses an overbooking strategy due to the considerable utilization improvements seen over guaranteed bidding. As the density of the workload increases the improvement gained by using reservations is greater than that of computing substitutes. The total allocation rates of 95.82 percent for the batch model, 90.32 percent for the low workload, and 78.34 percent for the medium workload represent lower allocation rate than that of using substitutes, however in the dense high workload the improvement exceeds the gains made using substitutes peaking at 55.75 percent of all tasks allocated.

6.3.5 Reservations, Substitutes, and Overbooking
The final configuration combines the use of reservations with the ability to compute substitute providers and overbook resources. This combination gives the best results for each of the workloads, with 100 percent of low workload tasks allocated, 98.38 percent of the medium workload allocated, 65.54 percent of high workload allocated, and 99.68 percent of the batch workload allocated. In the low and medium workload no auctions fail as providers are not fully utilized for long periods of time. Only one contract is rejected in the batch model and 75.67 contracts are rejected in the medium workload due to short periods of fully utilized providers. As the high workload consistently exceeds the capacity of the testbed, 65 percent of tasks allocated represents a very high degree of workload allocation, close to the maximum obtainable allocation.

6.3.6 Summary
Table 2 summarizes the performance of the high utilization strategies when applied to each of the workloads. The baseline guaranteed strategy is characterized by a very low allocation rate and poor average system utilization for each workload. Overbidding provides the single greatest increase in allocation and utilization across each workload, however the increased contract rejection rate may be viewed as pure overhead due to the wasted negotiation process. The use of second chance substitute providers greatly increases the overall allocation rate on the dynamic workloads by up to 27 percent therefore reducing allocation overhead and improving utilization.

The additional flexibility obtained when using reservations is advantageous to providers, especially in the most dense high workload. In fact, in the high workload the increased allocation rate obtained through reservations (31 percent) outperforms that of the substitute strategy (20 percent). However, it is unclear how many real-world tasks have the flexibility to be executed within a time window. Finally, the use of all four strategies simultaneously provides the highest allocation rate (100, 98.38, 65.54, 99.68 percent) and system utilization (71.72, 39.34, 68.91, 70.19 percent) over the testbed (for the low, medium, high, and batch workloads, respectively). This equates to substantial allocation improvement over a guaranteed approach of 190.64, 267.81, 286.44, and 109.03 percent for each of the workloads considered.

The improvements gained in the batch model are less dramatic for a number of reasons including increased provider capacity, utilization characteristics of the workload,
and decreased auction latency. The increased provider capacity significantly alters the allocation model from that used in the dynamic workloads. In this model, each provider effectively has 15 times the capacity, this increased capacity masks the limitations of overbidding as providers can win multiple simultaneous auctions without rejecting any agreements. In addition, the maximum system requirements of the workload only marginally exceed the capacity of the testbed resulting in a very high achievable allocation rate. The allocation rates for each strategy on the batch model are therefore similar to the low-utilization dynamic workload.

The results presented in this section demonstrate the value of employing such strategies to increase occupancy and utilization, the benefits are applicable to both consumers and providers using either batch or dynamic usage models.

6.3.7 Just-in-Time Bidding

JIT bidding is proposed as a means of reducing the effect of auction latency. The increased allocation rate due to JIT bidding are shown in Figs. 8 and 9 for the medium and high utilization workloads, respectively. The low utilization workload and batch model are not shown as the allocation rate is near optimal using the other strategies. For both the medium and high workload the number of tasks allocated increases by approximately 10 percent for each strategy up until a point of saturation—at which time not all bids are received before the auction closes. The two strategies employing second chance substitutes in both workloads do not exhibit as much of an improvement, as auctions will not fail as long as alternative substitutes are available. Table 3 shows the average number of substitutes considered for each strategy as JIT bidding gets closer to the auction close. Although the utilization improvements are smaller for the second chance strategies, the number of substitutes considered decreases as bidding occurs closer to the auction close. This is an additional benefit as it reduces the overhead required to compute substitute providers.

6.4 Economic Analysis

This section evaluates the implications of the high utilization strategies on provider revenue and also the impact of economic policies on the high utilization strategies themselves. The following experiments are conducted on the same virtualized testbed described in Section 6.2. In each

![Fig. 8. JIT bidding for the medium workload.](image1)

![Fig. 9. JIT bidding for the high workload.](image2)
TABLE 3
Average Number of Second Chance Substitute Providers Considered at Different JIT Deadlines

<table>
<thead>
<tr>
<th>Time Before Auction Close</th>
<th>S + O Average Depth</th>
<th>R + S + O Average Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 seconds</td>
<td>2.29</td>
<td>2.04</td>
</tr>
<tr>
<td>10 seconds</td>
<td>1.81</td>
<td>1.69</td>
</tr>
<tr>
<td>3 seconds</td>
<td>1.68</td>
<td>1.49</td>
</tr>
<tr>
<td>0.5 seconds</td>
<td>1.13</td>
<td>1.23</td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 seconds</td>
<td>2.55</td>
<td>2.76</td>
</tr>
<tr>
<td>10 seconds</td>
<td>1.98</td>
<td>2.16</td>
</tr>
<tr>
<td>3 seconds</td>
<td>1.77</td>
<td>1.74</td>
</tr>
<tr>
<td>0.5 seconds</td>
<td>1.19</td>
<td>1.29</td>
</tr>
</tbody>
</table>

experiment providers bid using the nonuniform pricing and penalty models described in Section 5. Bids are based on the requirements of a job (in units), where the price per unit is calculated (0 to 20). In this section, results are presented for the medium and high workloads only, details of the low and batch workloads are presented in [28].

6.4.1 Allocation

Fig. 10 shows the allocation rate when different pricing functions are used under each utilization strategy. The guaranteed (G) strategy results in near identical utilization across all pricing functions because every bid is a guarantee and therefore the bid value is inconsequential. The substitute methods (S + O and R + S + O) also produce near equal allocation independent of the pricing function used, this is because substitute methods are not bid sensitive—assuming there is sufficient available capacity rejected contracts can be substituted by a different provider at an increased cost. This highlights an additional advantage of using substitute strategies in an economic market.

The more risk averse pricing functions, Time and Win/Loss, produce lower allocation for the nonsubstitute strategies (O and R + O). This is primarily due to the way in which providers compute bids, increased contract rejection without substitutes, and the fact that all bidders employ the same strategy. Using these pricing functions, providers compute bids based on previous events, therefore a providers bid cannot increase until they win a new auction. As a result, from the time a bidder becomes the lowest bidder until they win another auction their bid will remain the lowest. During this time if multiple auctions are held there will be considerable failure.

Available capacity pricing also leads to slightly lower allocation rates when using reservations (R + O) on both workloads, this is because the calculation of utilization does not include queued reservations. Therefore, the bid price is reflective of currently running tasks only, in the situation where a larger task is queued the bid price will be artificially low. The constant pricing functions produce the same allocation rate as the other functions due to the random order in which auctions are advertised and ties are resolved.

6.4.2 Revenue

Provider revenue for each strategy and pricing function is shown in Fig. 11. As expected, revenue is closely related to the allocation rate presented in Fig. 10. The use of overbooking and substitute strategies (S + O and R + S + O) provides significant additional revenue over the guaranteed strategy for each pricing function considered, this increase is in line with the increased allocation rate. Interestingly the strategies without substitutes (O and R + O) exhibit lower revenue for the time-based pricing functions. This is due both to lower allocation rate and also the follow on effect from the poor

Fig. 10. Number of tasks allocated for each of the pricing functions and strategies considered.

Fig. 11. Provider revenue for each of the pricing functions and strategies considered.

Fig. 10. Number of tasks allocated for each of the pricing functions and strategies considered.
allocation rate—prices are kept artificially low as providers continue to lower prices over time.

### 6.4.3 Penalty Revenue

This section presents the revenue generated, for each strategy, when different penalty functions are applied. Results are only shown for two strategies (O and \( R + S + O \)) on the medium and high workloads. Complete analysis of the other strategies can be found in [28].

The penalty functions are denoted: Constant 200 (C200), Job Units (JU), Win Price (WP), Substitute Price (SP), Bid Difference (BD), Bid Difference/Depth 1 (BDD1). It should be noted that Bid Difference/Depth 2 (BDD2) is not shown as the overall system impact is equivalent to BD, the difference in these penalty functions is only noticeable when analyzing revenue for individual providers (shown in [28]). A No Penalty (NP) function is included for comparison to show the total revenue generated when no penalties are enforced.

The following figures show total revenue generated across all providers when each penalty function is applied. Positive revenue indicates a profit to the provider whereas negative revenue indicates a loss.

**Overbooking.** Fig. 12 shows the total revenue generated when providers employ only an overbooking strategy. For both workloads the system operates at a loss when including penalties for Time and Win/Loss(10:1) pricing functions, this is due both to the decreased allocation rate and low revenue outlined in the previous sections. The penalties imposed are significant as a large number of contracts are rejected by providers (e.g., 7,000 contracts are rejected using the Time(5) pricing function on the high workload).

In most cases, the revenue generated with penalties increases with the increase in revenue without penalties (NP). However, this is not the case for the WP penalty. In the high workload, WP penalties result in a significant loss for all pricing functions, this is reflective of the increased winning bid price of the different pricing functions. These results demonstrate that the combination of an inflexible pricing function and a naive overbooking strategy can result in substantial loss for providers. Plain overbooking strategies are therefore considered economically risky.

**Overbooking and second chance substitutes.** The use of second chance substitutes is proposed as a means of reducing the impact of contract violations on allocation rate caused by overbooking. From an economic standpoint this same approach can be used to minimize the penalties imposed. When supporting second chance substitute providers there are two different types of penalties that can be enforced: 1) contracts are rejected and are unable to be satisfied by a substitute provider, 2) contracts are rejected and then allocated to a substitute provider. The following analysis is designed to examine the second case when using substitute providers. Therefore, in the following results no penalty is applied if contracts are rejected without the ability to substitute the winner, this provides a basis to accurately compare the different penalty functions. When examining the revenue generated it should be noted that in reality additional penalties would be applied, however these would be constant across all penalty functions. These penalties would also be considerably less than the other strategies as the contract rejection rate is significantly lower (Table 2).

The total revenue with penalties using a substitute strategy is shown in Fig. 13. This figure shows that the penalty functions vary in their sensitivity to different pricing functions. For most pricing functions there is little difference between win price and substitute price penalties, which implies there is little bid variance between providers. However, in both workloads the Time and Random pricing
functions generate higher revenue using Win Price penalties rather than Substitute Price penalties, indicating a large spread between valuations. This difference is also reflected in the revenue drop due to BD penalties.

7 Conclusions

The utility model employed by commercial cloud providers has remodeled the need for efficient and responsive economic resource allocation in high-performance computing environments. While economic resource allocation provides a well studied and efficient means of scalable decentralized allocation it has been stereotyped as a low-performance solution due to the resource commitment overhead and latency in the allocation process. The high utilization strategies proposed in this paper are designed to minimize the impact of these factors to increase occupancy and improve system utilization.

The high utilization strategies have each been implemented in the DRIVE metascheduler and evaluated using a series of batch and interactive workloads designed to model different scenarios, including multiple high throughput, short job duration workloads in which auction mechanisms typically perform poorly. The individual strategies, and the combination of the different strategies, was shown to dramatically improve occupancy and utilization in a high-performance situation. The increase in allocation rate was shown to be up to 286 percent for the dynamic workloads and 109 percent for the batch model.

In addition to occupancy and utilization improvements these strategies also provide advantages under differing economic conditions. For example, the use of substitute providers was shown to be more price agnostic than other strategies due to the decreased allocation rate when a linear bidding strategy is used. Provider revenue also increased with the use of the proposed strategies, in part due to the increased allocation rate obtained. Finally, the effect of penalties on total revenue was shown to be heavily dependent on the penalty function used. The bid difference penalty, which represents the impact of the contract breach, resulted in only a small loss of total revenue across all providers. These results highlight that while these strategies can dramatically improve allocation performance, participants must fully consider the negative effects of the strategy used and associated penalty functions in order to optimize revenue.

Acknowledgments

The AuverGrid traces were kindly provided by the AuverGrid team (special thanks to Dr. Emmanuel Medernach), the owners of the AuverGrid system.

References


Kyle Chard received the BSc (Hons.) degree in computer science, the BSc degree in mathematics and electronics, and the PhD degree in computer science from Victoria University of Wellington in 2011. He is a senior researcher at the Computation Institute, University of Chicago and Argonne National Laboratory. His research interests include distributed meta-scheduling, Grid and cloud computing, economic resource allocation, social computing, services computing, and medical natural language processing. He is a member of the IEEE.

Kris Bubendorfer received the PhD degree, on mobile agent middleware, in computer science from the Victoria University of Wellington in 2002. He is the program director for Networking Engineering and senior lecturer in the school of Engineering and Computer Science at Victoria University of Wellington. His research interests include market oriented utility computing, social computing, digital provenance, and reputation. He is a member of the IEEE.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.