ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 2, February 2013

EFFICIENT RESOURCE ALLOCATION STRATEGIES FOR CLOUD DATA CENTERS

Jeyaram.G[1], Vidhya.V[2]

Faculty of CSE, Annai Vailankanni College of Engineering, Kanyakumari, India [1]

Faculty of CSE, Annai Vailankanni College of Engineering, Kanyakumari, India [2]

ABSTRACT: **Cloud computing has become a new age technology in enterprises and markets. Clouds allow access to applications and associated data from anywhere. Companies are able to rent resources from cloud for storage and other computational purposes so that their infrastructure cost can be reduced significantly. Further they can make use of company-wide access to applications, based on pay-as-you-go model. Hence there is no need for getting licenses for individual products. However one of the major pitfalls in cloud computing is related to optimizing the resources being allocated. Because of the uniqueness of the model, resource allocation is performed with the objective of minimizing the costs associated with it. The other challenges of resource allocation are meeting customer demands and application requirements. The performance limitations of existing economic allocation models are analysed 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 meta scheduler and evaluated with respect to Grid and cloud deployments.**

Index Terms: Resource allocation, cloud computing, grid computing, cloud services.

I. INTRODUCTION

CLOUD computing helps consumers to 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 like posted price but the system moves towards more sophisticated mechanisms, such as spot-pricing. In near future 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 advertized as a means of allocating resources in both centralized and decentralized computing systems [1]. An advantage of such system is allocation efficiency, scalability, clear incentives, and well-understood mechanisms as advantages. The system also includes poor performance, high latency, and high overheads.

Overheads in the sense, in 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, these inefficiencies are addressed by two general principles: 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 meta scheduler [2].

II. RELATED WORK

The earliest computational market enabled users to bid for compute time on a shared departmental

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 2, February 2013

machine. DRIVE, the system used for the experimental work in this paper, is one example of such a federated meta scheduler and is designed around the idea of "infrastructure free" secure cooperative markets. Another prominent example is Inter Cloud [8] which features a generic market model to match requests with providers using different negotiation protocols. Another alternative approach is spot pricing.

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.

Globus Architecture for Reservation and Allocation (GARA) [12] was one of the first projects to define basic advanced reservation architecture to support QoS reservations over heterogeneous resources. Many other schedulers have been developed so reservation aware schedulers have been shown to improve system utilization due to the additional flexibility specified by some consumers, [15].

Various papers have looked at last minute bidding and "sniping" [16], [17]. Typical motivation for last minute bidding is to combat "shill bidders" (fake bidders raising the price) and incremental bidding (bidding in increments rather than bidding ones true value or proxy bidding). JIT bidding for sealed bid auctions was first proposed in some earlier work [18] as a means of reducing the effect of auction latency in distributed auctions.

III. OPPORTUNITIES AND HIGH UTILIZATION STRATEGIES

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.

A. *Preauction*

1. Overbooking: Overbooking has been shown to provide substantial utilization and profit advantages [10] due to "no shows" 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]. Overbooking policies are carefully created and are generally based on historical data. Due to the widespread adoption of overbooking techniques in commercial domains there is substantial economic theory underpinning appropriate strategy [22], [23].

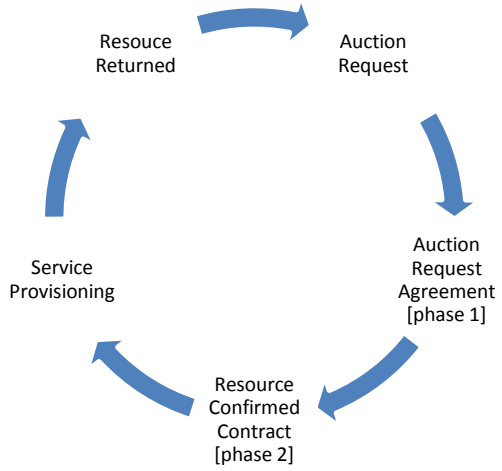

Fig 1: Reverse auction life cycle in DRIVE

B. *During Auction*

1. Just-in-Time Bidding: During negotiation it is possible that resource state may change and therefore invalidate a provider's 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.

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

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 2, February 2013

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. 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. This increase flexibility to substantial performance improvements for providers.

C. Post auction

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

1. Providers can reserve resources for the duration of the negotiation immediately,
2. They can wait for the result of the allocation before reservation.

Both the case 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).

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.

IV. 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. 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).

V. EXPERIMENTAL ECONOMY

The pricing functions used 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.

A. 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_{unit} is the price per job unit and b denotes a bid in the specified range $(b \in (0, B))$. Job units (JU) are defined as the product of CPU utilization and duration ($J_{units} = J_{utilization} \_ J_{duration}$). The Random and Constant pricing functions are baseline functions.

Random: the unit price is determined irrespective ofany other factors

$$P_{unit} = Random(0, B)$$

Constant: the unit price is the same for every request

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 2, February 2013

$$P_{unit} = c, c \in (0, B)$$

Available capacity: the unit price is calculated based on projected provider capacity at the time when the job would execute. $U_{provider}$ is the projected utilization of the provider, U_{job} is the utilization of the requested job, and $C_{provider}$ is the total capacity of the provider

$$P_{unit} = \left| \frac{U_{provider} + U_{job}}{C_{provider}} \times B \right|$$

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

$$P_{unit} = (RW - L) \times \frac{B}{R} + \frac{B}{2}$$

Time based: the unit price is based on the time since the provider last won an auction. The unit price decrements every T_{period} seconds, T_{last} win is the time since the last allocation. T_{last} win is set to 0 at time t_0

$$P_{unit} = B - \left(\frac{T_{lastwin}}{T_{period}} \right)$$

B. Penalty Functions

The penalty functions are divided into two distinct penalty types:

1. Constant penalties are fixed penalties that are statically defined irrespective of any other factors.
2. 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: α penalties are based on the relative size of the job or the established price, whereas β penalties attempt to model the increased cost incurred by the consumer using a ratio of the original and substitute prices. β penalties are only possible when second chance substitute providers are used. Specifically, the different penalty functions are:

Constant: a constant penalty defined statically irrespective of the job requirements or bid price

$$P_{defaulter} = c, c \in IR_{\geq 0}$$

Job units: an α penalty based on the requirements of the job in units. J_{units} is the number of units in a job, and c is a constant penalty per unit

$$P_{default} = J_{units} \times c, c \in IR_{\geq 0}$$

Win price (WP): an α penalty based on the winning bid (pre-substitutes). $Price_{win}$ is the price to be paid by the winning bidder:

$$P_{default} = \Pr ice_{win}$$

Substitute price: a α penalty based on the substitute bid. $Price_{substitute}$ is the price to be paid by the substitute winning bidder:

$$P_{default} = \Pr ice_{substitute}$$

Bid difference (BD): a β penalty defined as the difference between the original win price and the substitute price

$$P_{default} = \Pr ice_{substitute} - \Pr ice_{win}$$

Bid difference/depth: a β 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. $Depth_{substitute}$ is the number of substitutes considered

$$P_{default} = \frac{\Pr ice_{substitute} - \Pr ice_{win}}{Depth_{substitute}}$$

$$\forall_i \in D : P_i = \frac{\Pr ice_{substitute} - \Pr ice_{win}}{Depth_{substitute}}$$

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 2, February 2013

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.

VI. EVALUATION

Each of the strategies is evaluated with respect to allocation occupancy, utilization, and economic implications. Occupancy is defined as the number of contracts satisfied and utilization as the amount of a host's resource capacity that is used.

A. Synthetic Workloads

Several synthetic workl oad that generates different workload conditions are developed. 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: a sample can be taken over a fixed period of time to simulate a workload characterized by long duration batch processing 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. 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.

1. Batch Model: 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 favourable auction conditions as the ratio of auction latency to interarrival time is large. Reducing the sample size such that this workload can be hosted on our 20 machine test bed is impossible as the resulting number of jobs would be minimal. Instead, experiments using the batch workload utilize an increased test bed capacity by simulating "larger" providers.

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. To simulate this type of high performance dynamic model reduce the time based attributes of the workload by a factor of 1,000. By reducing each parameter equally relativity between parameters is maintained and therefore the distribution is not affected.

B. 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 agigabit Ethernet network. The machines each have Core2 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.

C. *Strategy Evaluation*

The strategies 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

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 2, February 2013

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.

R + S + O: Providers bid based on projected utilization at the time of job execution. In the event that a contract cannot be satisfied in the reservation window (even after moving other tasks), a losing provider may be substituted with a second chance provider.

In each experiment tasks from the workload trace are submitted to DRIVE for allocation. For each task DRIVE conducts an 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.

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. Rejections occur during auctioning and no contracts are rejected, as no provider should ever bid outside its means. 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. 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.

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 as providers only reach close to maximum capacity for a short period of time.

However, the number of contracts unable to be established directly effects system performance as the auction and contract negotiation processes are wasted.

3. Reservations and Overbooking: In the AuverGrid workload trace, there is no explicit execution window, 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. 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

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 2, February 2013

by using reservations is greater than that of computing substitutes

4. Reservations, Substitutes, and Overbooking: The final configuration combines the use of reservations with the ability to compute substitute providers and overbook resources leading to increase in successful bidding.

5. Just-in-Time Bidding

JIT bidding is proposed as a means of reducing the effect of auction latency. The increased allocation rates due to JIT bidding are shown in Figs. 2 and 3 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. 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.

VII. CONCLUSION

The utility model employed by commercial cloud providers has remotivated 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 meta scheduler 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, were shown to dramatically improve occupancy and utilization in a high performance situation.

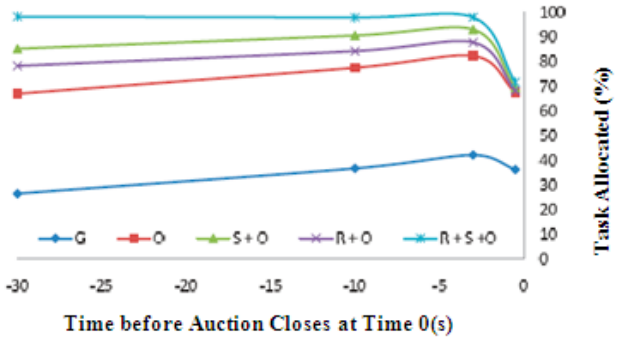

Fig: 2 JIT bidding for the medium workload

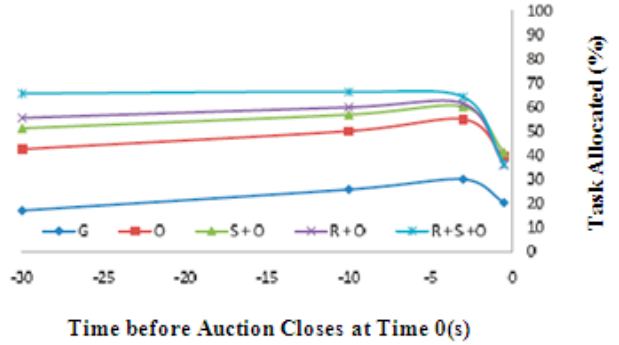

Fig: 3 JIT bidding for the high workload

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

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 2, February 2013

improve allocation performance, participants must fully consider the negative effects of the strategy used and associated penalty functions in order to optimize revenue.

ACKNOWLEDGEMENT

We thank our parents for their full support and continuous encouragement without whom we could not have achieved this level. The authors would like to thank the anonymous reviewers for their comments that helped to improve the quality of this paper. The authors are solely responsible for the views expressed in this paper.

REFERENCES

[1] I.E. Sutherland, "A Futures Market in Computer Time," Comm.ACM, vol. 11, no. 6, pp. 449-451, 1968.
[2] K. Chard and K. Bubendorfer, "Using Secure Auctions to Build A Distributed Meta-Scheduler for the Grid," Market Oriented Grid and Utility Computing, series Wiley Series on Parallel and Distributed Computing, R. Buyya and K. Bubendorfer, eds., pp. 569-588, Wiley, 2009.
[3] K. Chard, K. Bubendorfer, and P. Komisarczuk, "High Occupancy Resource Allocation for Grid and Cloud Systems, a Study With Drive," Proc. 19th ACM Int'l Symp. High Performance Distributed Computing (HPDC '10). pp. 73-84, 2010,
[4] C.A. Waldspurger, T. Hogg, B.A. Huberman, J.O. Kephart, and W.S. Stornetta, "Spawn: A Distributed Computational Economy," IEEE Trans. Software Eng., vol. 18, no. 2, pp. 103-117, Feb. 1992.
[5] T.W. Malone, R.E. Fikes, K.R. Grant, and M.T. Howard, "Enterprise: A Market-Like Task Scheduler for Distributed Computing Environments," The Ecology of Computation, pp. 177- 205, Elsevier Science Publishers (North-Holland), 1988.
[6] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/g: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid," Proc. Fourth Int'l Conf. High Performance Computing in Asia-Pacific Region (HPC Asia '00), pp. 283-289, 2000.
[7] D. Neumann, J. Sto¨ßer, A. Anandasivam, and N. Borissov, "Sorma - Building an Open Grid Market for Grid Resource Allocation," Proc. Fourth Int'l Workshop Grid Economics and Business Models (GECON '07), pp. 194-200, 2007.
[8] R. Buyya, R. Ranjan, and R.N. Calheiros, "Intercloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services," Proc. 10th Int'l Conf. Algorithms and Architectures for Parallel Processing, p. 20, 2010.
[9] M. Mattessa, C. Vecchiola, and R. Buyya, "Managing Peak Loads by Leasing Cloud Infrastructure Services from a Spot Market," Proc. 12th IEEE Int'l Conf. High Performance Computing and Comm. (HPCC '10), pp. 1-3, Sept. 2010.
[10] A. Sulistio, K.H. Kim, and R. Buyya, "Managing Cancellations and No-Shows of Reservations with Overbooking to Increase Resource Revenue," Proc. IEEE Eighth Int'l Symp. Cluster Computing and the Grid (CCGRID '08) pp. 267-276, 2008,
[11] G. Birkenheuer, A. Brinkmann, and H. Karl, "The Gain of Overbooking," Proc. 14th Int'l Workshop Job Scheduling Strategies for Parallel Proc. (JSSPP), pp. 80-100, 2009.
[12] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy, "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-allocation," Proc. Seventh Int'l Workshop Quality of Service (IWQoS '99), pp. 27-36, 1999,
[13] "Catalina Scheduler," www.sdsc.edu/catalina/, Jan. 2012.
[14] Sun Microsystems. Sun Grid Engine. http://gridengine. sunsource.net/, Jan. 2012.
[15] M.A. Netto, K. Bubendorfer, and R. Buyya, "Sla-Based Advance Reservations with Flexible and Adaptive Time Qos Parameters,"Proc. Fifth Int'l Conf. Service-Oriented Computing (ICSOC '07), pp. 119-131, 2007.
[16] A.E. Roth and A. Ockenfels, "Last-Minute Bidding And the Rulesfor Ending Second-Price Auctions: Evidence from Ebay and Amazon Auctions on the Internet," Am. Economic Rev., vol. 92,no. 4, pp. 1093-1103, 2002.
[17] P. Bajari and A. Hortacsu, "Economic Insights from InternetAuctions," J. Economic Literature, vol. 42, pp. 457-486, 2004.
[18] K. Bubendorfer, "Fine Grained Resource Reservation in Open Grid Economies," Proc. IEEE Second Int'l Conf. e-Science and Grid Computing (E-SCIENCE '06), p. 81, 2006.
[19] B.C. Smith, J.F. Leimkuhler, and R.M. Darrow, "Yield Managementat American Airlines," INTERFACES, vol. 22, no. 1, pp. 8-31, 1992.
[20] Y. Suzuki, "An Empirical Analysis of the Optimal Overbooking Policies for Us Major Airlines," Transportation Research Part E:Logistics and Transportation Rev., vol. 38, no. 2, pp. 135-149, 2002.
[21] R. Ball, M. Clement, F. Huang, Q. Snell, and C. Deccio,"Aggressive Telecommunications Overbooking Ratios," Proc. IEEE 23rd Int'l Conf. Performance, Computing, and Comm. (IPCCC),pp. 31-38, 2004.
[22] C. Chiu and C. Tsao, "The Optimal Airline Overbooking Strategy under Uncertainties," Proc. Eighth Int'l Conf. Knowledge-Based Intelligent Information and Eng. Systems (KES '04), pp. 937-945, 2004.
[23] J. Subramanian, S. Stidham Jr., and C.J. Lautenbacher, "Airline Yield Management with Overbooking, Cancellations, and No- Shows," Transportation Science, vol. 33, no. 2, pp. 147-167, 1999.
[24] C. Castillo, G.N. Rouskas, and K. Harfoush, "Efficient Resource Management Using Advance Reservations for Heterogeneous Grids," Proc. IEEE 22nd Int'l Symp. Parallel and Distributed Proc.(IPDPS '08), pp. 1-12, Apr. 2008.
[25] K. Chard and K. Bubendorfer, "A Distributed Economic Meta-Scheduler for the Grid," Proc. IEEE Eighth Int'l Symp. Cluster Computing and the Grid (CCGRID '08), pp. 542-547, 2008.
[26] K. Bubendorfer, B. Palmer, and I. Welch, "Trust and Privacy in Grid Resource Auctions," Encyclopedia of Grid Computing Technologies and Applications, E. Udoh and F. Wang, eds., IGI Global,
[27] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D.H.J. Epema, "The Grid Workloads Archive," Future Generation Computer Systems, vol. 24, no. 7, pp. 672-686, 2008.
[28] K. Chard, "Drive: A Distributed Economic Meta-Scheduler for the Federation of Grid and Cloud Systems," PhD. dissertation, School of Eng. and Computer Science, Victoria Univ. of Wellington, 2011.