A Comparison Between Mechanisms for Sequential Compute Resource Auctions

Andrew Byde HP Labs
Filton Road, Stoke Gifford
Bristol, BS34 8QZ
andrew.byde@hp.com

ABSTRACT

This paper describes simulations designed to test the relative efficiency of two different sequential auction mechanisms for allocating compute resources between users in a shared data-center. Specifically we model the environment of a data center dedicated to CGI rendering in which animators delegate responsibility for acquiring adequate compute resources to bidding agents that automously bid on their behalf. For each of two possible auction types we apply a genetic algorithm to a broad class of bidding strategies to determine a near-optimal bidding strategy for a specified auction type, and use statistics of the performance of these strategies to determine the most suitable auction type for this domain.

Categories and Subject Descriptors

I.6.3 [**Simulation and Modelling**]: Applications—*Economics*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems

1. INTRODUCTION

This paper describes simulations designed to test the relative efficiency of two different sequential auction mechanisms for allocating compute resources between users in a shared data-center. Specifically we model the environment of a data center dedicated to CGI rendering in which animators delegate responsibility for acquiring adequate compute resources to bidding agents that automously bid on their behalf. For each of two possible auction types we apply a genetic algorithm to a broad class of bidding strategies to determine a near-optimal bidding strategy for a specified auction type, and use statistics of the performance of these strategies to determine the most suitable auction type for this domain.

The business scenario we address is an **Inter-business** scenario of resource contention – in which the market mediates between different businesses that want access to the same underlying resources – as opposed to an **intra-business** scenario, in which a corporation uses "the market" as an organizing principle for the allocaiton of resources between competing groups within its own organization.

This paper is an attempt to assess the pros and cons of two different market mechanisms relative to a fair-share benchmark for the inter-business scenario[1], from the perspective of simulation. Specifically, we simulate a resource utility used by several competing agents, each of which has a collection of CGI frames to render within a certain time period and within a certain budget. To ground the simulations, we have based the model on the SE3D project [6], in which a select group of up-and-coming film-makers was given access to a state-of-the-art CGI rendering utility owned by HP, and were forced to acquire resources via markets. Section 3 describes the simulations in detail, including the model used for the rendering jobs on which the agents work, and the market mechanisms used. In Section 4 we describe the space of bidding strategies studied, and the optimization process applied to them (which depends upon the business scenario). Section 5 lays out the results of the various simulation and GA runs; Section 2 describes the relevance of this work to past work in the area of markets for compute resources, and we conclude in Section 6.

The main contribution of this paper is an analysis of how the operational properties of each of the market mechanisms studied affects the performance of the system as a whole, and how that relationship is affected by the design of model parameters such as the size and number of render jobs that agents have to complete.

2. RELATED WORK

The work of Ferguson et al on distributed markets for compute resources [5] is some of the earliest. Like [12], [4], [13] and [7], among others, allocation in Ferguson's system is done at the level of the individual server, in order to manage distributed resources. The benefit of this approach is that by distributing the mechanism a central point of control (and hence potential failure) is eliminated. The downside of a distributed mechanism is that allocations might be far from optimal, and might have unstable dynamics, depending on the coordination mechanisms used to migrate a job between nodes on the basis of pricing[2]. In any case, since this paper concerns a datacenter in which all compute resources are

[1]Further work will address the other business scenario and comparisons between the two.
[2]Ferguson gets around this problem by charging jobs to migrate, so that budget limitations damp migration.

centrally hosted, there is no benefit to using a distributed mechanism.

The other important reason for introducing markets (cited in, for example, [10], [3], [2]) is to ensure value driven quality of service. The utility of introducing a market for the resources allocation problem is thus to drive the selection of which jobs get scarce resources from an assessment of which jobs are most valuable (or, equivalently, least costly). The implicit design goal is therefore to maximize the aggregate utility of the user community, as expressed via their willingness to part with (or conserve) hard cash. This is the motivation behind SE3D [6].

The purpose of these simulation studies was to look beyond markets in which people bid directly to an environment in which agents act on users behalf to bid for resources. There is a large body of work in the agent systems community that deals with issues of this sort. A good example is the Trading Agent Competition [11], run each year since 2002 as a forum for investigating different autonomous agent strategies in a complex e-commerce task. See also [2], [8].

A variety of auction mechanisms have been tried as the basis for a market-based resource allocation system. In particular the Proportional Share and Generalized Vickrey mechanisms we use here have both been used before – for example PS in [4] and [7], GV in [12] and [13]. In addition other mechanisms have been used such as an ascending N^{th} price mechanism (in [9]) similar to GV but in which the price paid is the same for all winners and equal to the highest losing bid; and a first price mechanism in [5], [10]. Almost all the work cited evaluates a particular mechanism rather than comparing alternative market mechanisms[3].

3. RENDER UTILITY SIMULATOR

3.1 Infrastructure Model

The compute cluster is abstracted as a set of servers that are centrally allocated, identical with respect to capability, and resource constrained only with respect to the rate at which they can do work. We are not concerned here with memory, storage, network connectivity or physical location. For security reasons, servers in SE3D are not shared between animators, who have exclusive access to any server they have been allocated. The implication of this for the simulation is that the number of servers allocated to each agent is an integer.

3.2 Rendering Model

To simulate the challenges that an agent might be expected to face we have to simulate the operational details of the rendering application acting on a "realistic" payload. The basic unit of rendering is a frame, which is defined by its intrinsic work content, measured in server milliseconds. When a frame is assigned to a server, work begins on the frame, and proceeds either until the relevant number of milliseconds has elapsed, or until rendering on the frame is terminated. To be faithful to the SE3D environment the rendering simulator obeys the following constraints:

1. Work already done on a frame cannot be check-pointed, so that if rendering on a frame is terminated before the frame is complete, all work so far done on the frame is lost, and must be done again.
2. A frame is not parallelizable at all, so that for a single frame two servers would take the same amount of time as one.

The combination of these two properties makes performance non-linear with respect to resource levels, especially if frame workloads are large compared to the timescale of the auctions.

The task that each agent has to work on is a **render job**, consisting of a set of frames and constraints on how those frames can be rendered. For there to be any benefit from parallelization it must be possible to render more than one frame concurrently; for simplicity we assume total independence of frames so that frames may be rendered in any order, and any subset of frames may be rendered simultaneously on different servers. The constraints under which the set of frames must be rendered are:

1. Rendering may not commence before a specified start time (e.g. the time at which render source code is uploaded to the utility).
2. Rendering stops when a specified deadline is reached, and any frames in progress are terminated.
3. Resources for rendering must be purchased within a budget. Agents cannot spend more than their budget, and cease activity when their money runs out.

3.3 Auction Models

In SE3D, resources were sold on a rolling basis both for immediate use and in advance of later use. In these simulations we restricted attention to auctions for resources available immediately; further work will examine the extension to reservation markets, for which reasoning procedures will necessarily have to be more complex. Thus time is divided into "rounds". In each round there are three steps: first bids are gathered from each agents; then an auction is run to determine resource allocations until the next round; then work is done on each job with the resulting number of resources, according to the rendering model described above in Section 3.2.

There were two auction mechanisms tested in the SE3D project, and these were faithfully reproduced in the simulation.

3.3.1 Proportional Share Auction

In a Proportional Share (PS) auction for N servers, each agent submits a bid b^a; the number of resources allocated to each agent a is as close as possible to the share

$$N^a_{lim} = N \frac{b^a}{\sum_{a'} b^{a'}}. \qquad (1)$$

[3]The aparent exception, [5] is misleading because a first-price sealed-bid auction and a Dutch auction are economically equivalent. The difference in performance observed in [5] is due to the sub-optimality of the bidding algorithms.

Since this number is not in general an integer, we cannot in general give exactly N^a_{lim} servers (which are indivisible). Therefore the actual number N^a allocated is determined in a time-dependent way such that $|N^a - N^a_{lim}| \leq 1$ and such that if all bids remain the same for multiple successive rounds in the auction, then the average allocation $\langle N^a \rangle$ will converge[4] on N^a_{lim}. Before processing, bids are truncated to lie in $[0, B^a]$, where B^a is agent a's total remaining balance.

3.3.2 Generalized Vickrey Auction

In a Generalized Vickrey (GV) auction for N servers, each agent submits a vector of bids, $(b^a(1), b^a(2), \dots, b^a(N))$. The number $b^a(j)$ represents the maximum amount of money that agent a would be willing to pay to secure exactly j resources. The resource allocation N^a to each agent is calculated so as to maximize the sum of maximum prices $S(N) = \sum_a b^a(N^a)$. The price paid by each agent is set to be the negative impact that the presence of that agent has on this total as calculated over the other agents: with agent a removed we calculate the allocation $N^{a'}_{-a}$ for the other agents a' that maximizes $S_{-a}(N_{-a}) = \sum_{a' \neq a} b^{a'}(N^{a'}_{-a})$. The price paid by agent a is

$$S_{-a}(N_{-a}) - (S(N) - b^a(A_a)) = \sum_{a' \neq a} b^{a'}(N^{a'}_{-a}) - b^{a'}(N^{a'}). \tag{2}$$

If there are several allocations that give rise to the same maximal total $S(N)$, we cycle between those that minimize the difference between allocations to different agents. For example, if two agents bid $(10, 20, 30)$ for a total of 3 resources, any allocation of all the resources has value 30, the two allocations that minimize the difference between agents are $(1, 2)$ and $(2, 1)$ and so we cycle between these. The period of this cycle in time steps is a parameter of the allocation mechanism[4]. As with the PS auction, bids $b^a(n)$ are set to zero when they are negative and to B^a when greater than the agent's remaining funds.

The GV mechanism is obviously more complicated to understand (as a bidder) than the proportional share mechanism. Its potential benefit is that it is known to have desirable game-theoretic properties: In a one-shot auction the GV mechanism is incentive compatible – i.e. a rational agent's best strategy is to truthfully reveal their demand for each given number of resources. GV should allocate resources to those agents who truly need the resources most, maximising the total utility of all agents combined. Of course in common with most humans, animators are not likely to be "rational" in the sense that economists intend it, and from either a human or bidding agent standpoint, the calculation of the marginal value of resources in a single auction round is extremely complex since it depends on expected success in future rounds.

In SE3D, to simplify bidding in the GV auction so that bidders were not compelled to enter a complete vector of bids for each time step, we constrained them to use bid vectors from a 3-parameter family, parameterized by the maximum number of units for which they wanted to bid (N_{max}), a total price for all units ($b_{max} = b^a(N_{max})$), and a risk factor (r) in $[-1, 1]$ which determines the rate of decrease of marginal demand $b^a(n) - b^a(n-1)$. For a full description of the bidding functions see [1].

[4]In practice, extra resources cycled around once every three time steps, to avoid too much "thrashing"

4. BIDDING STRATEGIES

4.1 Measuring Bidding Strategy Performance

The performance of a bidding strategy can only be measured indirectly from the performance of an agent using it to render a particular sequence of frames against opposing agents with their own jobs to work on. There are three core metrics for measuring an agent's performance on a job:

1. The number of frames remaining to be rendered when the deadline was reached;
2. The amount of money remaining when the agents deadline was reached; and
3. The amount of work remaining when the agents deadline was reached.

An agent's performance depends on its bidding strategy, the job it has to complete (some jobs are intrinsically easier than others), the number of resources available, and the agents against which it is competing for resources – both their strategies and their jobs. Except in trivial cases the set of all possible jobs with which an agent might be faced is too large to enumerate exhaustively. To evaluate the performance of a bidding strategy we therefore run a large number of simulations with jobs for all participants selected at random from some suitable distribution. From this we infer the average performance of a bidding strategy in the context of a collection of opposing strategies. Two strategies can be compared to one another by running the above process twice – once for each strategy – in which case we use the same sequence of jobs for each, so as to reduce the variance of the comparison with respect to random job selection.

4.2 Restricting the space of strategies

In its most general form, an agent's bid depends on the current state of its job, the list of past bids it has made and resulting allocations and prices, and any accumulated history of previous runs. In cases of interesting scale the space of strategies is therefore far too large to enumerate. However, we can make the problem of identifying good strategies tractable by only considering agents that use some of the information available, and by constraining them to use it in certain ways.

When choosing their bid functions, we only allow agents to consider static job description data such as the job's budget, start time, deadline and number of frames, and dynamic job progress data such as the current time, the number of frames completed and workload in each, and the total funds remaining. Agents were carried no knowledge over from one run to another, except that when no frames had been completed, agents based their predictions on a fixed parameter, whose calibration can be seen as inter-run learning. The job workload prediction algorithm was simple: each frame left to render was expected to have work content equal to the average work content in all frames rendered to date.

4.3 PS Polynomial Family of Strategies

Agents operating in the PS market used a strategy whose bid value was determined as a rational polynomial function of the current time and time to complete (t, d). Specifically

$$b^a/B^a = (c^a_1/d + c^a_2 + c^a_3.d + c^a_4.d^2) \times (1 + c^a_5.t + c^a_6.t^2), \quad (3)$$

where B^a is agent a's remaining funds at the time of bidding. A given strategy within this family is determined by specifying the 6 values above. The default strategy was $b^a/B^a = 2/d$, which allocates an equal amount of remaining budget over all remaining auction rounds.

4.4 GV Polynomial Family of Strategies

Agents operating in the GV auction bid using the same constrained family of bid functions with which animators were confronted. They used a strategy whose parameters (N_{max}, b_{max}, r) were determined as rational polynomial functions of the current time and time to complete (t, d). Specifically

$$\begin{aligned} N_{max}/N &= (c^N_1/d + c^N_2 + c^N_3.d + c^N_4.d^2) \\ &\qquad \times(1 + c^N_5.t + c^N_6.t^2), \\ b_{max}/B^a &= (c^b_1/d + c^b_2 + c^b_3.d + c^b_4.d^2) \\ &\qquad \times(1 + c^b_5.t + c^b_6.t^2), \\ r &= (c^r_1/d + c^r_2 + c^r_3.d + c^r_4.d^2) \\ &\qquad \times(1 + c^r_5.t + c^r_6.t^2), \end{aligned} \quad (4)$$

where N is the total number of resources available in the auction. As with PS strategies, the data are truncated to their appropriate ranges. A given strategy within this family is determined by specifying the 18 values above. The default strategy was $N_{max}/N = 1$, $b_{max}/B^a = 1/d$, $r = 0$, spreading the total budget evenly over the time interval as before.

It is clear that even within the restricted families described above we have no hope of finding any equilibria analytically. When confronted with this situation, different researchers have taken different approaches. We chose to apply a genetic algorithm to the sets of parameter values in (3) and (4), in order to evolve appropriate strategies.

5. RESULTS

The standard simulation environment consists of 6 agents, 15 resources, job start time from $U[0, 10]$, job durations from $U[20, 40]$, number of frames from $U[10, 20]$, and budget per frame from $N(0.8, 0.2)$ – the normal distribution with mean 0.8 and standard deviation 0.2; these parameters are chosen to maximise the difference in performance between the bidding algorithms, by avoiding scenarios that are either too easy or too hard.

A benchmark resource allocation mechanism is "fair share" (FS) in which each "active" agent is simply given an equal share[4] of the resources; agents are active if the start time of their job has been reached, the deadline has not been reached, and the job is not complete.

5.1 Evolved Polynomial Bid Functions

The pattern is clear: the evolved GV strategy commits far more resources to short-deadline late-in-the-day states than the default strategy. One possible reason for this is that if the deadline is large then it is probable that the agent's job will outlast the opposition anyway, and by bidding less early on, other agents with tighter deadlines (who are willing to spend more) will get their jobs done in time to allow later processing at lower cost. Note that the values are far above 1, showing that agents bid at a potentially unsustainable level. However, in the GV mechanism agents usually do not pay their full bid amount, so the high level of b_{max} can also be interpretted as an implicit calculation of likely payments for given bid data.

Other parameters from the GV bid function are not shown, to save space, but are also interesting. The trend is towards risk aversion as deadlines approach, and towards bidding for more units (up to 3 times as many) as are available in the auction! Clearly the bid function is using the parameters to discover bid vector shapes that were not anticipated in the design of the parameter space.

The evolved PS agent is much closer to its default value, with neither function significantly different from (in the sense that 1 lies within the population envelope).

5.2 Performance data

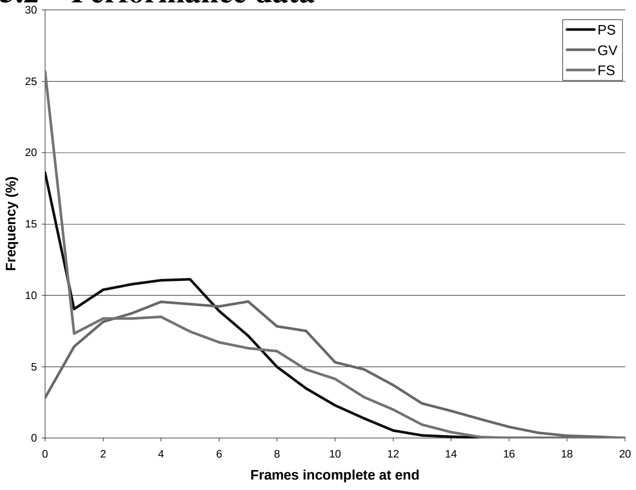

Figure 1: Distribution of number of incomplete frames.

Figure 1 shows the distribution of the number of frames that have not been rendered by the time of the jobs deadline. Although FS has the greatest frequency of runs in which all frames are completed 25.7%, it has a higher proportion of cases in which a large number of frames is incomplete than PS. This is reflected in the standard deviation of frames complete – 5.54 for FS as opposed to 3.85 for PS. In fact PS has both the lowest variance and the lowest mean: 3.85 for PS as opposed to 4.90 for FS. GV has the highest mean 6.38 and the highest standard deviation 7.43.

To try to analyse why mechanisms might give the performance metrics they do, we can examine the statistics of resource allocation.

Figure 2 shows allocation distributions for each allocation mechanism at time step 15. As can be seen, Fair share

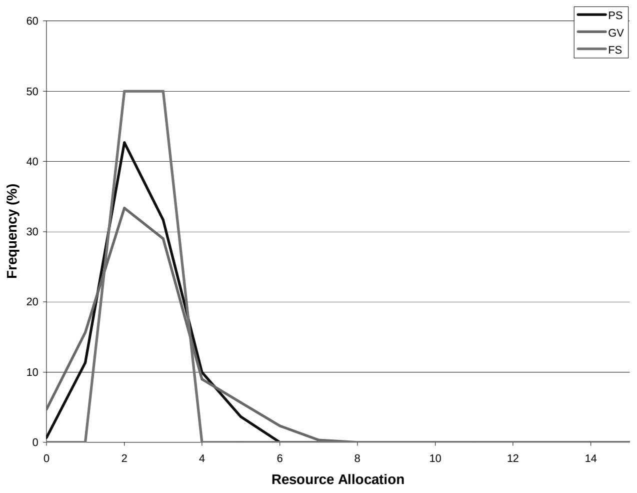

Figure 2: Distribution of resource allocation for PS and GV markets at time 15.

has the most predictable distribution of allocations at this time. By time 15 all agents' jobs have started and none have ended, so all agents are active. Since there are 6 agents and 15 resources, each agent gets either 2 or 3 with 50% probability. For PS, and GV, the range of possible allocations at this time is wider, resulting not only from the fact that job priority, as represented via budget, is taken into account, but also from the intrinsic volatility of each mechanism itself. In this respect PS is seen to be lead to a lower spread of resource allocations (for the same set of jobs) than GV.

The data in figure 2 indicates that the three mechanisms lead to quite different degrees of variability of allocation. However, it only speaks directly to variability between runs at a given time, rather than variability for a single run over the lifetime of a job. For this we should look at the distribution of all absolute changes to resource allocation.

The standard deviations of allocation change are 1.12, 1.14 and 1.24 for PS, FS and GV respectively, showing that although the allocations generated by FS have markedly lower variance than the other mechanisms at time 15, over the lifetime of a job FS gives a similar although slightly higher level of variability to PS. Interestingly PS is far ahead when it comes to the number of time steps in which allocation did not change at all: 75%, as opposed to 66% and 67% for GV and FS respectively.

6. CONCLUSION

Markets as an organizing principal for large compute clusters have been proposed for several reasons. In distributed systems with no central resource allocation mechanism they are a natural fit for the problem of incenting self-interested participants to behave well. In this paper we examine a different scenario, in which the provider of a centralized compute resource uses markets to mediate the conflicting needs of a variety of users having access to a shared resource. While it might seem that the choice of mechanism for such a market is arbitrary on the grounds that supply and demand should lead to the same equilibrium, in fact we find that when supply and demand fluctuate and are uncertain, and when allocation changes are costly, the precise dynamics of a mechanism can greatly affect the efficiency of the whole system. An inappropriate mechanism – GV in this case – can be worse than simply sharing the resources equally among participants irrespective of need; an appropriate mechanism however can consistently do better for the community as a whole.

7. REFERENCES

[1] A. Byde. A comparison between mechanisms for sequential compute resource auctions. In *Proceedings of* 5^{th} *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 06)*, 2006.

[2] A. Byde, M. Sallé, and C. Bartolini. Market-based resource allocation for utility data centers. Technical report, HP Labs, 2003. HPL-2003-188.

[3] J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centres. In *Symposium on Operating Systems Principles*, pages 103–116, 2001.

[4] B. N. Chun and D. E. Culler. Market-based proportional resource sharing for clusters. Technical report, University of California at Berkeley, Berkeley, CA, USA, 2000.

[5] D. Ferguson, Y. Yemini, and C. Nikolau. Microeconomic algorithms for load balancing in distributed systems. In *Proc. of the* 8^{th} *International Conference on Distributed Computer Systems*, pages 491–499. IEEE, 1988.

[6] H. Labs. SE3D project. http://www.hpl.hp.com/SE3D.

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

[8] C. Preist and M. van Tol. Adaptive agents in a persistent shout double auction. Technical report, HP Labs, 2003. HPL-2003-242.

[9] D. M. Reeves, M. P. Wellman, J. K. MacKie-Mason, and A. Osepayshvili. Exploring bidding strategies for market-based scheduling. *Decision Support Systems*, 39:67–85, mar 2005.

[10] I. Stoica, H. Abdel-Wahab, and A. Pothen. A microeconomic scheduler for parallel computers. In *Proc IPPS '95 Workshop on Job Scheduling Strategies for Parallel Processing*, pages 122–135, apr 1995.

[11] Trading agent competition. http://www.sics.se/tac.

[12] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. SPAWN: A distributed computational economy. *Software Engineering*, 18(2):103–117, 1992.

[13] A. A. Young, B. N. Chun, A. C. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures. In *Proc. of the* 1^{st} *Workshop on Operating System and Architectural Support for the on demand IT Infrastructure*, oct 2004.