

A Comparison Between Mechanisms for Sequential Compute Resource Auctions

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

General Terms

Economics, Experimentation, Management

Keywords

Bidding, Utility Computing, GRID, Market-based Control, Genetic Algorithms

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. 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.

To ground the simulations, we have based our model on the SE3D project [3], 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. We hypothesise an extension to SE3D in which animators are given bidding agents to relieve them of the chore of bidding, and seek to understand what appropriate bidding strategies for those agents might be, and what impact market mechanism choice would have.

2. RELATED WORK

The work of Ferguson et al on distributed markets for compute resources [2] is some of the earliest. Like [1] and [4], among many 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. In any case, since this paper concerns a datacenter in which all compute resources are centrally hosted, there is no benefit to using a distributed mechanism.

Bidding to acquire complex sets of resources is not a novel area of study for agents: the canonical example in this field is of course the Trading Agent Competition [7]. Optimal strategy selection is notoriously dependent on the specific configuration of goods, mechanisms, preferences and so on. Our focus however is not so much on finding good strategies as it is in comparing different mechanisms in the domain of Utility Computing.

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], GV in [8]. In addition other mechanisms have been used such as an ascending N^{th} price mechanism (in [5]) 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 [2], [6]. Almost all the work cited evaluates a particular mechanism rather than comparing alternative market mechanisms.

3. RENDER UTILITY SIMULATOR

3.1 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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

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 start time, end time (deadline) and total budget.

3.2 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. 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 3.1.

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

3.2.1 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

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

Since this number is not in general an integer, we cannot in general give exactly N_{lim}^a servers (which are indivisible). Therefore the actual number N^a allocated is determined in a time-dependent way such that $|N^a - N_{lim}^a| \leq 1$ 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 converge¹ on N_{lim}^a . Before processing, bids are truncated to lie in $[0, B^a]$, where B^a is agent a 's total remaining balance.

3.2.2 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

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

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

¹In practice, extra resources cycled around once every three time steps, to avoid too much “thrashing”

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 mechanism¹.

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 $b^a(n) - b^a(n-1)$.

4. BIDDING STRATEGIES

4.1 Strategy Space

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.

We chose to restrict the space of bidding strategies by making the fraction of remaining budget bid, dependent only on current time step t and time to deadline d , with rational polynomial dependency. In the PS markets for example, each bidding strategy was defined by six constants $c_1, \dots, c_6$, via the function

$$b^a/B^a = (c_1^a/d + c_2^a + c_3^a \cdot d + c_4^a \cdot d^2) \times (1 + c_5^a \cdot t + c_6^a \cdot t^2), \quad (3)$$

where B^a is agent a 's remaining funds at the time of bidding.

Likewise in GV markets each of the three strategic parameters, total bid b_{max} , maximum number of units N_{max} , and risk factor r , were represented in this way, leading to an 18-dimensional bidding strategy space.

4.2 Strategy Optimization

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 order to evolve appropriate strategies.

To do this, a population of strategies was instantiated to multiple copies of the default strategy ($b/B = 1/d$ for PS markets, $b_{max}/B = 1/d$, $r = 1$, $N_{max} = N$ for GV). In each generation each individual was placed into a run of the simulation with randomly selected opponents drawn from the population; each agent was given a randomly generated job to perform, and the simulation run to determine the performance of the agents – and hence the strategies. After a number of such runs for each individual, the averaged performance data were used as a measure of the fitness of each individual. To generate the next generation, genomes were selected with preferential weighting given to high-fitness individuals, and a new set of parameters generated from the two parents by crossover. Mutation was applied to each parameter, and the resulting parameter-set inserted into the new population. Elitism was used, so that the best individual in a given generation was always transferred to the next generation.

5. RESULTS

The standard simulation environment consisted of 6 agents competing for 15 resources with job start time $\sim U[0, 10]$; time from start to deadline $\sim U[20, 40]$; number of frames $\sim U[10, 20]$; and budget per frame $\sim N(0.8, 0.2)$. Agent performance was measured by the number of frames completed by the job's deadline.

These parameters were chosen to give maximal performance difference between the mechanisms. When the system is under very little contention all agents get all their jobs done all the time; when the system is under very heavy contention, agents fail so often that their performance is not much affected by mechanism choice either. In between the mechanism makes a difference, and this was the parameter zone from which the experimental conditions were selected.

A benchmark resource allocation mechanism is "fair share" (FS) in which each active agent is simply given an equal share of the resources.

With the setup above, using FS, the mean number of frames incomplete at the end of a simulation run was 4.12 with a standard deviation of 5.54 – a significant number given that the total number of frames per job was in the range 10 – 20.

5.1 Performance data

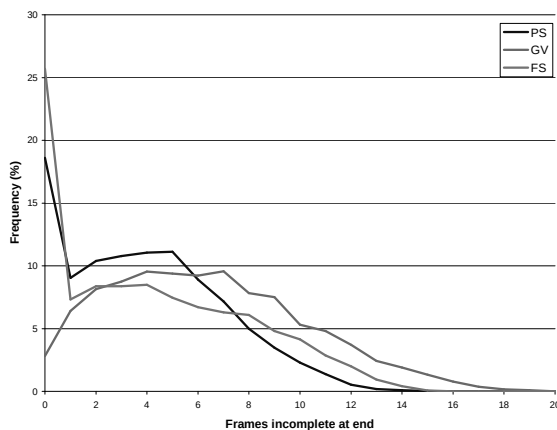


Figure 1: Distribution of number of incomplete frames.

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. We can examine the difference between runs at a given 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, giving a standard deviation of allocation of 0.5. For PS the standard deviation is 0.97, while for

GV it is 1.31, demonstrating that the allocations fluctuated more widely with the GV mechanism than PS, which itself varied more widely than FS.

Some of this variability is due to variability in job budget, start time etc. We can control for this by examining the change in allocation from time step to time step throughout the lifetime of each job. 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.

These relative standings with respect to volatility are the same when repeated for for allocation change measured relative to the average allocation given to a job over its lifetime. Here the standard deviations are 0.699 for PS and FS, and 0.881 for GV.

6. CONCLUSION

While it might seem that the choice of market mechanism through which to sell shared compute resources 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] 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.
- [2] D. Ferguson, Y. Yemini, and C. Nikolau. Microeconomic algorithms for load balancing in distributed systems. In *Proc. of the 8th International Conference on Distributed Computer Systems*, pages 491–499. IEEE, 1988.
- [3] H. Labs. SE3D project. <http://www.hpl.hp.com/SE3D>.
- [4] K. Lai, B. A. Huberman, and L. Fine. Tycoon: A distributed market-based resource allocation system, 2004.
- [5] 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.
- [6] 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.
- [7] Trading agent competition. <http://www.sics.se/tac>.
- [8] 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.