18^{th} World IMACS/MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

Why decide: is a user's estimation of job completion time useful in grid resource allocation?

Timothy M. Lynar, Ric D. Herbert, Simon, and William J. Chivers

Faculty of Science and Information Technology, The University of Newcastle, Ourimbah, NSW, 2258, Australia.
E-Mail: timothy.lynar@studentmail.newcastle.edu.au

There are many resource allocation mechanisms for grid computing. One trait that many allocation mechanisms have, is that they require the user to estimate how long their task will take to execute on a system. The result of the user's input can have a considerable impact on scheduling, and can affect the grids ability to meet quality of service requirements for other jobs. Incorrect estimates by users could result in other jobs being turned away that should not have, or jobs being accepted, that should not have been.

The user's estimation could conceivably be accurate if the user is running the same jobs repeatedly on the same hardware. However this is rarely the case in a grid environment. This estimation is often an estimation that is made with limited or no knowledge of how long a task will take on the underlying hardware and can be considered to be a guess.

In this paper we have tested the accuracy of user estimation of task execution, through simulating the grid environment. The simulation includes a simple agent-based batch auction to distribute jobs to resources.

The simulated environment contains a number of resources. Tasks are submitted to the environment periodically by agents who estimate the length of time their job will take. Each agent has only one job but submits that job several times throughout the simulation. Each job requires a random amount of processing and each resource can process a random amount per time-step.

There are three groups, each group with equal quantities of agents. Each group uses one of three strategies to estimate the length of time their task will take to execute on the grid. One strategy uses the agent's limited memory to estimate the length of execution based on the time previous tasks took for that agent to execute over the grid; a second strategy utilises a history from the assigned resource as to how long previous jobs have taken to execute on the resource; and the third strategy is a zero intelligence strategy, it is a random guess.

The results show whether there is any significant difference between the accuracy of an estimation made using historical data from a user, using historical data from the resource, or a random guess.

The results have implications for the design of grid resource allocation mechanisms, and for how users interact with current resource allocation mechanisms. It also raised the question, when should a user's input be required and on what will it be based?

1 INTRODUCTION

We are all familiar with the issue of determining how many resources our computing task is going to need and, in particular, how long the task will take to complete on a computer system. The problem can become even more difficult when we are submitting our task to a computational grid as we have little control over which computer nodes in the grid will be used to execute our job. We often do not even know what hardware is used in the grid. The task's time requirement estimation becomes even more difficult with heterogeneous nodes.

How the task is allocated across the nodes in the grid is the task of the grid's resource allocation mechanism. There are many and varied resource allocation mechanisms, but one trait that many have in common is that they require the user to estimate how long their task will take to execute. Incorrect estimates by users could result in a deterioration of quality of service indicators for the grid.

In this paper we investigate whether users can accurately estimate the time their task will take to execute on a grid of heterogenous computing resources. The approach we use is to build a simulation model of users repeatedly submitting tasks to the simulated grid. A resource allocation mechanism allocates the tasks and we compare the user's estimate with the actual time taken for the task to complete. User agents follow particular strategies in estimating their tasks' execution time. One strategy is a random guess and we compare how that strategy performs with strategies based upon experiential learning.

2 BACKGROUND

Computational grids are becoming increasing important. Basically computational grids are networks of distributed software and hardware resources that can be utilised to compute common tasks (Németh & Sunderam 2002). Foster & Karonis (1998) expand on the concept of computational grids, likening computational grids to the power grid, where users and resources are distributed and users have access to a dependable and consistent supply of computational power: "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" (Foster & Karonis 1998, p.3). Grids typically span many geographical locations and contain heterogeneous resources with varying computational abilities (Tarricone & Esposito 2004). Often grids are made of existing computing resources (e.g. research computing clusters and high performance computers) logically formed into a consolidated computing resource. A specific example is the world wide Large Hadron Collider (LHC) Grid which consists of 140 computing centres in 33 countries (lcg.web.cern.ch/).

There are many claimed potential benefits to grid computing including the ability to utilise an existing high performance computing environment for a low price (Foster & Karonis 1998). Supercomputers can solve problems in seconds that would otherwise take a personal computer days to solve, however the speed of a supercomputer comes at a high monetary cost. Schneck (1990) found that supercomputers are by far the most expensive form of computer. Grid computing offers one way of gaining the power and speed of a supercomputer for a fraction of the price.

A key component of the grid is the resource allocation mechanism which allocates user's tasks to computing resources. In many grid resource allocation mechanisms a user's tasks may be allocated to a variety of resources with no guarantee that the task will be distributed to the same resources each time it is executed. Further, grids may contain resources with vastly different processing abilities, including high-performance supercomputers and aging commodity workstations. This heterogeneous nature of grid computing can make it difficult to estimate the time a user's application may take to execute. For example, Zhang & Inoguchi (2006) discovered that although many performance prediction tools are accurate it is still impossible to produce an exact prediction in a grid environment.

Accurate estimation of task execution time is important as it can effect quality of service agreements which may stipulate deadlines and or response times (Spooner et al. 2003). Furthermore resource allocation mechanisms may drop partially completed tasks that over run their allotted processing time. Spooner et al.

(2003, p. 94) argue that "...performance prediction will be crucially important to prevent wasteful allocations and avoid committing to impractical user metrics".

The estimated time of execution is often used by resource allocation mechanisms to make resource allocation decisions. It is often assumed that the estimated time is accurate (Zhang & Inoguchi 2006). Numerous systems have been developed to estimate task execution time on heterogeneous grid resources using various techniques. Some estimation mechanisms utilise historical data (see, for example, Ali et al. 2000, Casanova et al. 2000, Smith et al. 2004, Kim et al. 2003); other mechanisms (such as Spooner et al. 2003) analyse the application to be executed to estimate its resource requirements; and other mechanisms (for example Gao et al. 2005, Sun & Wu 2005) utilise artificial intelligence methods.

There is a significant body of research on resource allocation and scheduling; many of these mechanisms rely on task execution time predictions, and many assume these predictions to be accurate. Ali et al. (2000) utilise an expected time to execute metrics which contains pre-processed estimates for all expected tasks on each computational resource. These values are typically assumed and can come from benchmarking, task profiling, prior executions or user input. The Casanova et al. (2000) scheduling system also assumes accurate estimates of execution time are provided to the system from past execution history, or input by the user. Smith et al. (2004) selectively utilise past execution history in combination with genetic algorithms to predict execution time of tasks. Kim et al. (2003) examine heuristic techniques of scheduling dynamically in heterogeneous computing environments. Execution times are calculated using a heuristic method known as gamma distribution. The Spooner et al. (2003) approach to estimating task execution time utilises descriptions of resources and modelling of tasks to produce estimations of application runtime.

In this paper we do not use a performance prediction tool, rather we examine the accuracy of strategies for user predictions of execution time. Our approach is to build a simulation model to investigate the accuracy of simple execution prediction time estimation strategies that users may implement. These strategies assume the user has limited knowledge.

3 THE MODEL

To test the importance of user's time estimation for tasks in a grid resource allocation system we have constructed a simulation model. The model contains users and a simulated grid.

The model is an agent-based model, written in Java and based upon the model of Herbert & Turton (2007). The model was executed on a standalone workstation. The key features of the model are:

Users There is a set, $\mathcal{U}$, of n user-agents who at each round, k, submit a single task, t_i, to be processed on the grid. Each user-agent has a strategy attribute, s_i, for estimating the time the grid will take to process the task.

Tasks There is a set, $\mathcal{T}$, of n tasks, one for each user-agent at each round. The task has attributes of processing effort required and estimated completion time. We allocate a random value for processing effort. The estimated completion time is determined by the user-agent's strategy.

Strategy Each user has a strategy attribute from the set of strategies, $\mathcal{S}$. This strategy is used by the user to estimate the time taken for their task to be completed. Strategy s_1 uses the agent's limited memory to estimate the length of execution based on the time tasks in previous rounds took for that agent to execute over the grid. The agent uses the average time from his *memory* of previous tasks. A second strategy, s_2, utilises the history of the assigned computing resource. The user submits the task and the resource allocator assigns it a resource and utilises a *history* from the assigned resource as to how long previous tasks have taken to execute on the resource. Again, the resource has a limited memory length and uses the average of it's memory. The third strategy, s_3, is a *zero intelligence* strategy, whereby the user agent makes a random guess. This strategy can be taken as the user not making any decision.

Grid The grid consists of resource node agents and a resource allocation agent.

Resources There is a set, $\mathcal{R}$, of m resource node agents. Each resource has an attribute of processing power, p_m, that can be completed in a round.

Algorithm 1 The algorithm for the resource allocating auction in round k.

```
Shuffle tasks 𝒯
Shuffle resources ℛ
for all Tasks 𝒯 do
  Retrieve the next task t_{i,k}
  for all Resources R do
    Retrieve the next resource r_{j,k}
    Assign task t_{j,k} to resource r_{i,k}
  end for
end for
Process tasks
```

Allocator There is a resource allocator agent that allocates tasks to resources. The result of the allocator agent is that for each round t_i is mapped to r_j. We use an economic based controller as a resource allocator. We consider tasks as asks and allow resources to bid for the asks. The allocator then uses an auction to match the asks and bids and hence allocate tasks to resources. For the results presented in this paper we use a batch auction.

In the results presented here we implement the model with the following:

- Each task is allocated a random processing requirement.
- The user has the same task for each round.
- Resources are randomly allocated a fixed processing power.
- Each resource can only work on one task per round.
- For the resource allocation mechanism we use a batch auction at each round. If there are more tasks than resources ($n > m$), then some tasks will not be processed in this round and are kept until the next round. Also, if there are less tasks than resources some resources will not be utilised. The resource allocator is given in Algorithm 1.

4 SIMULATION RESULTS

4.1 Our Approach

The focus of this paper is on whether a user can accurately estimate the time their task will take to execute on a grid of heterogenous computing resources. To examine this issue we have simulated user estimation, using three different user-agent estimation strategies. The output of these strategies is compared to the actual time it took for the task to execute. A one way analysis of variance is used on the difference between the agents' estimates and how long an agent took to execute a task. The results show if there is a significant difference between the strategies.

We test the hypotheses (H1) that: user-agent knowledge of past execution time of a task on will result in significantly more accurate estimates of execution time than user-agents with no knowledge of prior task execution time.

Our null hypothesis (H0) is that: none of the user-agent strategies examined will result in more accurate estimates than the zero intelligence strategy. That is, there is no significant difference in the results.

We examine the effect of changing parameters of the model including number of rounds, number of resource agents, number of user-agents, maximum resource processing power, maximum task processing requirements, and memory length of user and resource agents. Over the 25 sets the parameters of each value has been varied. The parameters utilised in each set is presented in Table 1.

Set	Number of rounds	Number of resources	Number of agents	Memory length	ANOVA P-value
	Benchmark				
0	100	1000	100	10	0.00
	Vary memory length				
1	100	1000	100	1	0.00
2	100	1000	100	3	0.00
3	100	1000	100	7	0.00
4	100	1000	100	30	0.00
5	100	1000	100	50	0.00
6	100	1000	100	70	0.00
7	100	1000	100	100	0.00
	Vary number of user-agents				
8	100	1000	200	10	0.00
9	100	1000	300	10	0.00
10	100	1000	50	10	0.00
11	100	1000	30	10	0.00
12	100	1000	10	10	0.00
13	100	1000	3	10	0.00
14	100	1000	1	10	0.00
	Vary number of resources				
15	100	3000	100	10	0.00
16	100	7000	100	10	0.00
17	100	10000	100	10	0.00
18	100	300	100	10	0.00
19	100	500	100	10	0.00
	Vary number of rounds				
20	300	1000	100	10	0.00
21	700	1000	100	10	0.00
22	1000	1000	100	10	0.00
23	50	1000	100	10	0.00
24	10	1000	100	10	0.00
25	1	1000	100	10	0.96

Table 1. Parameter sets and results

4.2 Analysis of Results

The null hypothesis H0 stated that: None of the strategies examined will result in more accurate estimates than the zero intelligence strategy. This has been rejected. In all but one set the zero intelligence strategy produced the least accurate estimates. An analysis of the results was performed to test the proposed hypothesis. The results are remarkably uniform, all sets bar one set showed that the two proposed strategies performed significantly better than the zero intelligence strategy. The only set where the zero intelligence strategy performed on par with either the history or the memory strategies was Set 25 (Figure 1); this set will be discussed later.

A one-way analysis of variance (ANOVA) shows whether or not there is a significant difference between the sets. However it does not show which sets are significantly different. To show which sets are different the Tukey-Kramer Honestly significant difference (HSD) test has been employed. Table 1 shows the ANOVA p-value, that is the probability that this differences between the sets was a a result of chance. The p-value of sets 0–24 is 0.00, this indicates that there is no likelihood that the difference is through chance, and the difference between the sets is therefore significant.

The differences between the user agents' estimates and the actual time taken to execute were examined. In Set 1 the history strategy scored a mean value of 3.3, the memory strategy scores a mean value of 3.8 and the

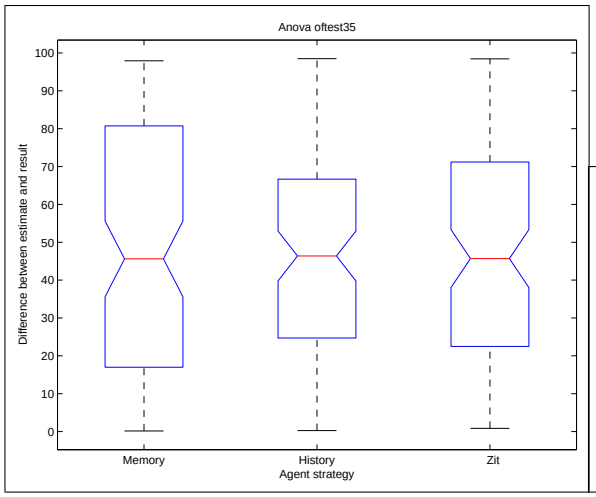

Figure 1. ANOVA Results of Set 25.

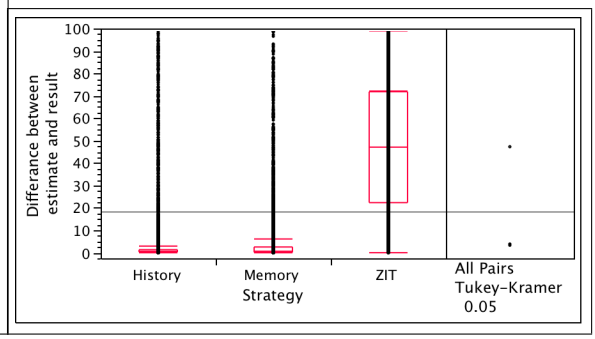

Figure 2. ANOVA Results of Set 1 with HSD.

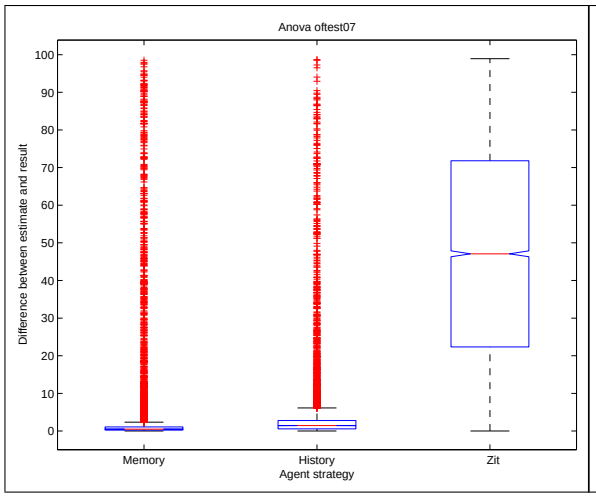

Figure 3. ANOVA Results of Set 7.

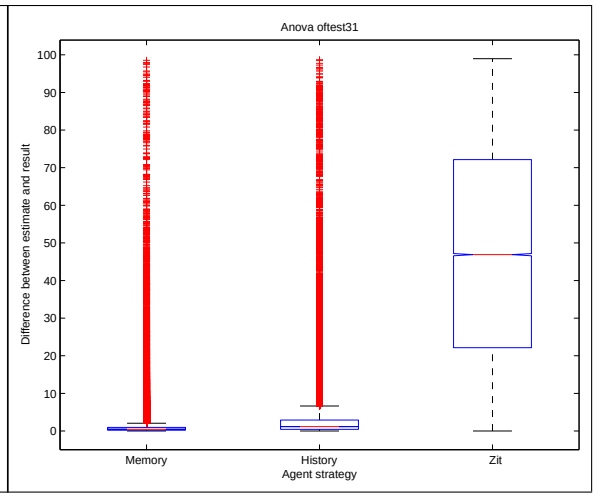

Figure 4. ANOVA Results of Set 21.

zero intelligence strategy scored 47.2 as its mean value. The ANOVA indicated that there was a significant difference between the sets. To show which sets were significantly different the Tukey-kramer HSD test was used. Figure 2 shows the results of the test. The overlapping circles on the far right of the figure indicate that the mean values of the history strategy and memory strategy are not significantly different from each other, but are both significantly different from that of the zero intelligence strategy. This result was the same for all sets except set 25.

Figures 3 and 4 show similar results to Figure 2, these results are consistent across all sets except Set 25, which shows less difference between the three strategies. Set 25 is unique in that there is only one round, so the competing strategies would have been equivalent to the zero intelligence strategy and would have made an initial zero intelligence bid.

Although the history and memory strategies offer a substantial improvement over the zero intelligence strategy, there are still a substantial number of estimates with both strategies that are over 80 time steps away from the actual execution time. These strategies do result in estimates that are on average between 3 and 4 time steps away from the actual execution time. This in turn has shown the null hypothesis to be false. The memory and the history strategy both consistently perform significantly better than that of the zero intelligence strategy.

5 CONCLUDING REMARKS

In this paper we have examined whether memory based user-agent strategies, were more accurate at estimating task execution time on a grid, than a zero intelligence approach. An agent-based model was created to simulate the grid, user-agents were assigned strategies which they utilised to estimate how long their task would need to execute. An analysis of user agent estimation strategies has shown that estimation based on past experience will result in closer estimates to actual execution time than could otherwise be achieved through guessing, even without any prior knowledge in a heterogeneous grid environment. The results show that in the majority of cases there was little difference between the memory and history based strategies. User estimation utilising a limited memory based strategy is more accurate than a zero intelligence approach.

REFERENCES

Ali, S., Siegel, H. J., Maheswaran, M., Hensgen, D. & Ali, S. (2000), 'Representing task and machine heterogeneities for heterogeneous computing systems', *Journal of Science and Engineering, Special 50 th Anniversary Issue* **3**, 195–207.

Casanova, H., Zagorodnov, D., Berman, F. & Legrand, A. (2000), Heuristics for scheduling parameter sweep applications in grid environments, *in* 'HCW '00: Proceedings of the 9th Heterogeneous Computing Workshop', IEEE Computer Society, Washington, DC, USA, p. 349.

Foster, I. & Karonis, N. T. (1998), A grid-enabled MPI: Message passing in heterogeneous distributed computing systems, *in* 'SC Conference', IEEE Computer Society, Los Alamitos, CA, USA, p. 46.

Gao, Y., Rong, H. & Huang, J. Z. (2005), 'Adaptive grid job scheduling with genetic algorithms', *Future Generation Computer Systems* **21**(1), 151 – 161.

Herbert, R. D. & Turton, P. (2007), Evolving strategies of agents in an auction model, *in* L. Oxley & D. Kulasiri, eds, 'MODSIM 2007 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand', MSANZ, Canberra, ACT, Australia, pp. 1103–1109.

Kim, J.-K., Shivle, S., Siegel, H. J., Maciejewski, A. A., Braun, T. D., Schneider, M., Tideman, S., Chitta, R., Dilmaghani, R. B., Joshi, R., Kaul, A., Sharma, A., Sripada, S., Vangari, P. & Yellampalli, S. S. (2003), Dynamic mapping in a heterogeneous environment with tasks having priorities and multiple deadlines, *in* 'IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing', IEEE Computer Society, Washington, DC, USA, p. 98.1.

Németh, Z. & Sunderam, V. (2002), *Computational Science ICCS 2002*, Vol. 2330/2002 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, chapter A Comparison of Conventional Distributed Computing Environments and Computational Grids, pp. 729–738.

Schneck, P. B. (1990), 'Supercomputers', *Annual Reviews Computer Science* **4**, 13–36.

Smith, W., Foster, I. & Taylor, V. (2004), 'Predicting application run times with historical information', *J. Parallel Distrib. Comput.* **64**(9), 1007–1016.

Spooner, D., Jarvis, S., Cao, J., Saini, S. & Nudd, G. (2003), 'Local grid scheduling techniques using performance prediction', *Computers and Digital Techniques, IEE Proceedings* - **150**(2), 87–96.

Sun, X.-H. & Wu, M. (2005), GHS: a performance system of grid computing, *in* 'Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International', p. 6 pp.

Tarricone, L. & Esposito, A. (2004), *Grid computing for electromagnetics*, Artech House, Boston.

Zhang, Y. & Inoguchi, Y. (2006), 'Influence of inaccurate performance prediction on task scheduling in a grid environment', *IEICE Transactions on Information and Systems* **E89-D**(2), 479–486.