A Time Optimization Algorithm for Scheduling Bag-of-Task Applications in Auction-based Proportional Share Systems

Anthony Sulistio and Rajkumar Buyya
Grid Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
ICT Building, 111 Barry Street, Carlton, VIC 3053
{anthony, raj}@cs.mu.oz.au

Abstract

Grid and peer-to-peer (P2P) network technologies enable aggregation of distributed resources for solving large-scale and computationally-intensive applications. These technologies are well-suited for Bag-of-Tasks (BoT) applications, because each application consists of many parallel and independent tasks. With multiple users competing for the same resources, the key challenge is to finish a user application within a specified deadline.

In this paper, we propose a time optimization algorithm that schedules a user application on auction-based resource allocation systems. These allocation systems, which are based on proportional share, allow users to bid higher in order to gain more resource shares. Therefore, this algorithm adjusts a user bid periodically on these systems in order to finish the application on time.

1. Introduction

Grid [7] and *peer-to-peer* (P2P) [10] network technologies enable aggregation of distributed resources for solving large-scale and computationally-intensive applications. These technologies are well-suited for Bag-of-Tasks (BoT) applications [5], because each application consists of many parallel and independent tasks. Some projects such as Nimrod-G [2], SETI@home [1] and MyGrid [6] utilize these technologies to schedule parameter-sweep or compute-intensive applications to available resources.

Resource allocation is a key challenge in a grid environment because resources can be part of one or more virtual organizations (VOs). The concept of a VO allows users and institutions to gain access to their accumulated pool of resources to run applications from a specific field [8], such as high-energy physics or aerospace design. As a result, users from different VOs are competing to use the same resources.

Maximizing utilization and ensuring fairness among users are two major issues in resource allocation. A system that uses a Proportional Share (PS) algorithm [12, 11], calculates a user share (percentage) of a resource based on a user weight in relation to the total weight of all users in the system. However, a PS allocation system does not check the validity of each user weight. This will lead to incorrect priority when one user gives a low-priority task the same weight as high-priority tasks of other users.

Combining a PS system with an auction model prevents the above problem, because the system allocates a resource share to users based on their bids, not weights. Therefore, users with a tight deadline will bid higher in order to gain more resource shares. In effect, this combination enforces users to disclose the importance of their task to the system.

The main drawback of the above approach is that users have to adjust their bidding price manually on each resource. A broker, who is acting on behalf of a user, can reduce a significant burden on the user by interacting with these resources automatically. Therefore, the user only needs to specify a Quality of Service (QoS) parameter such as deadline or budget to the broker.

The contribution of this paper is a time optimization algorithm of a broker in scheduling a BoT application in auction-based PS allocation systems. The aim of this algorithm is to minimize the total run time of a user application in order to satisfy a given deadline. The second objective is to reduce the total cost spent, only if the deadline can be reached. However, the deadline set by the user is not a hard or fixed deadline. It mainly acts as a guideline to the broker to adjust its bidding valuation dynamically, rather than use the same or maximum bid throughout.

We propose two variants of the time optimization algorithm. The first one considers submitting tasks to all avail-

able resources, including in other VO domains. On the other hand, the second variant of the algorithm submits tasks to local resources initially. As the deadline approaches, this algorithm then schedule the remaining tasks to resources in other VO domains. In this paper, we set the policy of each resource to earn a surcharge fee for accepting user tasks from other VO domains. Therefore, submitting tasks to local resources are cheaper than the global ones.

The rest of this paper is organized as follows: Section 2 mentions related work whereas Section 3 describes an overview of the model. Section 4 explains the two variations of the time optimization algorithm. These algorithms are then evaluated in Section 5. Finally, Section 6 concludes the paper and suggests some further work to be done.

2. Related Work

There are many systems that support proportional share (PS) allocation with an economy model, such as Spawn [14] and REXEC [4]. However, these systems are mainly for cluster computing environments. Tycoon [9], on the other hand, is intended to work on a grid environment. Therefore, our model follows a similar approach described in Tycoon and extends it to support multiple VO domains.

From a broker or user scheduling perspective, a Nimrod-G broker [2] and a Tycoon agent [9] are somewhat similar to what we propose in this paper.

In Nimrod-G, a user specifies QoS parameters, such as deadline and budget to a broker. Then, the broker schedules user tasks to resources with different allocation systems, which may not necessarily use a PS scheduler. However, the broker is not able to perform any bidding to these systems in order to gain more resource shares.

In Tycoon, a user specifies his/her preference of each resource by giving a weight to it manually. Then, the agent selects which resource to bid on based on the user weight and total bid of other users for each resource. The main objective of our work is to minimize the completion time of a user application. Therefore, a user does not need to specify a weight on each resource. In addition, our work is primarily focused on scheduling BoT applications across multiple VO domains. The Tycoon's agent does not recognize resources that are part of a VO domain.

3. Description of the Model

The model used in our evaluation is depicted in Figure 1, where one VO domain consists of a Grid Information Service (GIS) and one or more resources and users. Figure 1 also shows the interaction between relevant components in our model. The explanation of these interaction steps are explained below:

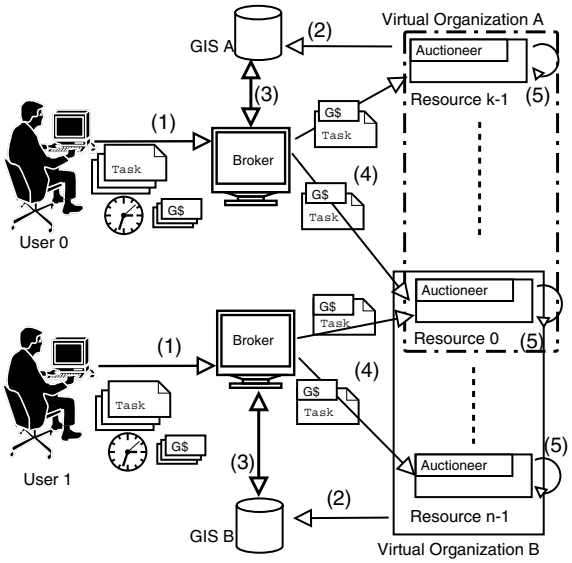

Figure 1. An overview of the model for two VO domains. Resource 0 is part of VO domain A and B.

1. User 0 sends to a broker his/her tasks and an initial fund with a specified deadline time. The same applies to User 1. In this model, the money is represented in Grid dollars (G$).

2. Each resource advertises its availability to a designated GIS. In Figure 1, Resource 0 is part of VO domain A and B. Hence, this resource registers to both GIS of domain A and B.

3. The broker queries a list of available resources to the GIS. In Figure 1, the broker of User 0 queries to the GIS of domain A, because it is running an application specific to domain A only. Likewise for User 1 running an application in domain B.

4. The broker submits tasks to these resources. In addition, the broker sends a periodic bid to each resource to increase / decrease a user share.

5. A resource re-computes each user share based on his/her bid at every bidding interval (see Section 3.2 for more details).

As mentioned previously, this model follows a similar approach described in Tycoon [9]. However, we extend the Tycoon's model into it to support multiple VO domains. The functionality of each component mentioned in Figure 1 is briefly described below. Detailed explanations of the Tycoon model can be found in [9].

3.1. Grid Information Service (GIS)

Each VO domain has a GIS that is responsible for storing information of available resources. The information given by a resource consists of a name, location, hardware specification, and a threshold for minimum and maximum bidding price. In multiple VO domains, a resource that belongs to a particular VO domain, will register to the relevant GIS. Hence, if a resource is part of n VO domains, then it will have to register to n GIS components.

In this model, a resource advertises to a GIS only once, unless the resource changes its information or decides not to be part of this domain anymore. Users contact the resource directly to find the latest update. This aims to reduce the communication overhead incurred by the GIS as mentioned in [9].

Users contact their GIS if they want to know the location and availability of local resources. A local resource means it is part of the same VO domain as the user. To find out information of global resources, the GIS will then contact other GIS components for their list of local resources. If the resources are registered in both local and other VO domains, then the GIS will omit them from the global resource list.

3.2. Resource

A resource r has an Auctioneer, which is responsible in accepting users bid and allocating CPU cycles to users according to their bids. At every period P, the Auctioneer re-calculates each user share and cost on r. In this model, memory and cache access are not explicitly represented, but they are implicitly captured in the task execution time.

We use the same formula as mentioned in [9] for granting resource share S to each user i over some period P, which is

$$S_i = \frac{B_i}{\sum_{k=0}^{n-1} B_k} C_r \tag{1}$$

and

$$B_i = \frac{lb_i}{tm_i}, \quad B_i \in [minB_r, maxB_r] \tag{2}$$

where B_i is the bidding price of i, lb_i is the local balance of i (in G$), tm_i is the bidding interval time of i, $minB_r$ is the minimum bidding price of r, $maxB_r$ is the maximum bidding price of r, and C_r is the total processing capability of r.

For example, according to Figure 1, User 0 and User 1 send their tasks to Resource 0. User 0 bids G$10 for 20 minutes, whereas User 1 bids G$6 for 20 minutes as well. Hence, according to eq. (1), Resource 0 allocates approximately 62.5% resource share to User 0 and 37.5% share to User 1.

Eq. (1) allows users to bid higher in order to gain more resource share. However, to prevent a race condition, where users continually out-bid each other at each period P, a constraint on minimum and maximum user bidding is imposed by r in eq. (2). Hence, this constraint prevents users from completely dominating resource shares or from manipulating the cost, such that it becomes too expensive.

The Auctioneer then calculates the cost for each user i, according to [9]

$$cost_i = min\left(\frac{q_i}{S_i},\ 1\right) B_i \tag{3}$$

where q_i is the amount of the resource that i actually consumes during P. This equation guarantees that if a user task consumes less than P, the Auctioneer only charges the time actually used, and not the whole period, to user i.

With our model, a resource has the following policies:

- the Auctioneer executes user tasks in a Time-Shared mode.
- the Auctioneer does not share a user bid information to other users, nor to different Auctioneers.
- the Auctioneer keeps using B_i until it is modified or cancelled by user i.
- the Auctioneer charges users after all their tasks have been completed.
- a user can only run one task at a time. Other user tasks will be put into a queue by the Auctioneer.
- a task cannot be pre-empted once it is in execution.
- the resource accepts tasks from other VO domains with an additional surcharge fee.
- if a different user bid arrives after a resource bidding interval P_n, then the Auctioneer will use it on P_{n+1}.

3.3. User

Each user has a broker that is responsible in monitoring the progress of a BoT application and managing on how much to bid on each resource. A user can be part of many VO domains. However, in this model, we restrict the user to only run one application of a specific VO domain. In addition, each user receives a fixed amount of money at a regular interval time.

We developed two variants of a time optimization algorithm of a broker, which aim to minimize the total run time of a user application in order to satisfy a given deadline. The detailed explanations of these algorithms are mentioned in the next section.

4. A Time Optimization Algorithm

In this section, two variants of a time optimization algorithm are illustrated. Before an experiment starts, each user i must specify to a broker a deadline D_i on how long a user application is expected to take to complete, and an initial fund. In addition, each user i must specify a time period P_i to enable the broker to adjust the user bid on a resource r based on its share. This allows users with a tight deadline to bid more frequently.

Let R be a set of n grid resources, and T be a set of m independent tasks, where each task has a variable length l. In case of a parameter-sweep application, all tasks have the same length.

4.1. A Global Domain Policy

The aim of this policy is to run a user application before the deadline D on all available resources, not only the ones from a local VO domain. Hence, for every period P time, this policy tries to make sure tasks on each resource can be completed before the deadline. The detailed explanation of this policy is described below.

4.1.1. Resource Discovery

A broker, who is acting on behalf of a user, contacts a GIS to get a list of available local and global resources. On each resource r, the broker gets the information of $\langle minBid_r, maxBid_r, C_r, PE_r \rangle$, where $minB_r$ is the minimum bidding price of r, $maxB_r$ is the maximum bidding price of r, C_r is the total processing capability of r, and PE_r is the total number of Processing Element (PE) or CPU on r.

4.1.2. Initial Bidding Price

The broker finds the slowest resource $r_{slowest} \in R$ by comparing its $C_{slowest}$ to C of other resources. Then, the broker tries to estimate on how long T would run on R. Assuming C has a unit of MIPS (Millions Instructions Per Second), hence, a task length l has a unit of MI (Millions Instructions).

Assuming for a worst case scenario, where some tasks are executed to only one PE in $r_{slowest}$, then the initial estimated completion time would be

$$est\_time = \frac{avgLength \times PE_{slowest}}{C_{slowest}}$$

with an average length of m submitted tasks to be

$$avgLength = \frac{\sum_{k=1}^{m} l_k}{m}$$

Next, the broker opens a new account on $\forall r \in R$. This account is needed by r to keep track of a current user bidding price, cost and total balance. Afterwards, the broker performs an initial bidding on r for the duration of P time

$$bid(r, P) = \begin{cases} maxBid_r * P, & \text{if } est\_time > D \\ avgBid_r * P, & \text{otherwise} \end{cases} \quad (4)$$

where $avgBid_r$ is the average bidding price of r. From equation (4), est_time can be seen as an indicator of whether a user has a tight deadline or not.

4.1.3. Tasks Submission

Each task $t \in T$ is submitted to one of available resources. Moreover, tasks with smaller length l are sent to less powerfull resources if they are able to meet the deadline. If all tasks have same l, then these resources will receive a smaller quantity. Then, the remaining tasks are to other resources by using a uniform distribution. If a resource fails, then tasks are rescheduled to a different resource.

4.1.4. Bidding Performance

For every period P time, the broker queries the current user progress on $\forall r \in R$. Each resource r gives the broker information that contains $\langle S_r, B_r, L_{total} \rangle$, where S_r is the current user share in MI, B_r is the current user bidding price, and L_{total} is the remaining total task length (including for all user tasks that are in the resource queue).

With the above information, the broker then calculates the task completion time on r, which is

$$finish\_time = \frac{L_{total}}{S_r} \quad (5)$$

We define $time$ to be the remaining experiment time, which is $time = (D + start\_time) - now$, where $start\_time$ and now representing the experiment start and current time respectively. The broker then face with 3 possible cases:

1. $(time \ / \ 2) < finish\_time < time$: keep using the same B_r, because tasks will be completed before D.
2. $finish\_time \leq (time \ / \ 2)$: decrease B_r, because tasks will be completed much earlier than D.
3. $finish\_time \geq time$: increase B_r, because tasks will be finished later than the specified D.

For case (2) and (3), how much a new bidding price B_{r_new} the broker is willing to increase / decrease from B_r, is decided by

$$B_{r\_new} = \begin{cases} \dfrac{finish\_time}{time} B_r, & \text{if } time > 0 \\ maxBid_r, & \text{if } D \text{ has passed} \end{cases} \quad (6)$$

with a constraint of $B_{r\_new} \in [minBid_r, maxBid_r]$. Therefore, a broker submits a new bid to a resource r over a P period of time, with

$$bid(r, P) = \begin{cases} B_r * P, & \text{for case (1)} \\ B_{r\_new} * P, & \text{for case (2) and (3)} \end{cases}$$

4.2. A Local Domain Policy

The aim of this policy is to run a user application before the deadline D time preferably on local resources only, because they do not incur a surcharge fee. Hence, for every P period of time, this policy tries to make sure tasks on each local resource can be completed before the deadline time. If D is approaching soon, then the remaining tasks may be submitted to global resources if necessary.

This policy has a similar approach to the Global Domain (GD) Policy discussed previously. Hence, differences of the two variants are highlighted below.

Let R_l be a set of u local resources, and R_g be a set of v global resources from other VO domains. The broker submits κ number of tasks to each resource $r \in R_l$ by using a uniform distribution. We define $\kappa = m \div n$, where m is the total number of tasks, and n is the total number of resources (including local and global ones). Then the remaining tasks are stored into $List$.

For every P period of time, the broker queries the current user progress on $\forall r \in R$. Global resources that do not schedule users tasks are ignored for the time being.

Identical to the GD policy, the broker determines $finish\_time$, $time$, B_{r_new}, and $bid(r, P)$ on a resource r. However, this policy will also try to submit one or more remaining tasks $t \in List$ to r if there is a sufficient time.

We define L_{task} to be the sum of tasks length that can be scheduled on r, and $L_{task} \geq 0$. As a result, equation (5) is modified to

$$finish\_time = \frac{L_{total} + L_{task}}{S_r}$$

Finally, if there are still tasks $t \in List$ and the current progress is more than half of D, then submit all tasks to best available resources by using a uniform distribution. Available resources in this context means local or global resources, that currently have none or smaller number of user tasks stored in the queue than others.

Name	Location	Resource Types	PE	C
R_0	GRIDS Lab, Univ. of Melbourne	dual Intel Xeon 2.6 Ghz	4	1050
R_1	Dept. of Physics, Univ. of Sydney	dual Intel Xeon 2.6 Ghz	4	1050
R_2	Dept. of Computer Sc., Univ. of Adelaide	dual Intel Xeon 2.6 Ghz	4	1050
R_3	Australia National Univ., Canberra	dual Intel Xeon 2.6 Ghz	4	1050
R_4	Dept. of Physics, Univ. of Melbourne	PC with Intel P4 2.0 Ghz	1	684

Table 1. Australian Belle Analysis Data Grid testbed resources simulated using GridSim.

5. Performance Evaluation

In this section, we evaluate the two policies discussed earlier to determine their effectiveness in scheduling a user application with a given deadline. A No Optimization (No-Opt) policy is also introduced in the comparison.

A user with No-Opt policy submits tasks uniformly to all available resources and puts the same bid throughout the whole time. Hence, with this policy, all resources earns an amount equal to the user bid.

We carried out the performance evaluation of these policies by using simulation. Therefore, we implemented the model described in Section 3 using the GridSim [3] simulation toolkit. This allows us to consider different administration policies at each resource that would otherwise be difficult to implement in real Grid testbeds.

5.1. Simulation Setup

The simulated grid environment consists of five resources $\{R_0, R_1, ..., R_4\}$ as summarized in Table 1. These resources are part of Belle Analysis Data Grid (BADG) testbed in Australia to analyze high-energy physics experiment data [15]. In the BADG testbed, all resources have the same 70GB harddisk capacity. With respect to a memory specification, $R_0 - R_3$ have 2GB RAM each, whereas R_4 only has 512MB RAM.

In GridSim, total processing capability of a resource C is modeled in the form of MIPS (Million Instructions Per Second) as devised by Standard Performance Evaluation Corporation (SPEC) [13]. Hence, a task length is measured in Millions Instructions (MI) unit. We use the rating of a SPEC CPU200 integer benchmark that measures the performance of processors and memory in a system.

An auction-based proportional share (PS) scheduler is used by all resources, with a bidding interval of every 60 seconds. The slowest resource, i.e. R_4 as described in Ta-

User Name	VO Domain	Start Time (min)	Bid Price (cents / min)	Opt. Policy
U_0	A	1	20	No-Opt
U_1	A	1	20	No-Opt
U_2	B	3	varies	global
U_3	B	3	varies	local
U_4	B	3	40	No-Opt
U_5	B	5	50	No-Opt
U_6	A	8	40	No-Opt
U_7	B	10	varies	global
U_8	A	10	varies	local
U_9	A	10	23	No-Opt

Table 2. Users participating in two VO domains.

ble 1, charges users [1 ... 100] cents per minute. Other resources, which are more powerful, cost [5 ... 500] cents per minute. In addition, a resource charges an extra 20% of a user bid for users that are from other VO domains.

The following simulation setups are also carried out:

- 10 created users $\{U_0, U_1, \ ..., \ U_9\}$, where U_2 and U_7 use a broker with a Global Domain (global) policy. U_3 and U_8 use a broker with a Local Domain (local) policy. The rest uses a broker with a No-Opt policy.
- each user has an initial fund of G$100 with a deadline time of 10 hours.
- each user has the same bidding interval P of 10 minutes.
- each user runs an application that is part of one VO domain only.
- each user has a BoT application that consists of 50 tasks, where a task length l is uniformly distributed in [700,000 ... 800,000] Millions Instructions (MI) unit.
- 2 new users start their experiments at every two minutes interval by using a Poisson distribution.

5.2. Scenario

In this experiment, there are two VO domains: A and B, where resources R_1 and R_3 are part of domain A and R_0, R_2 and R_4 are part of domain B. Table 2 shows which VO domain the users run their application. In addition, each user starting time, bid price and policy are also shown. Users with a No-Opt policy perform static bidding, i.e. putting the same bid price throughout the whole time for all resources.

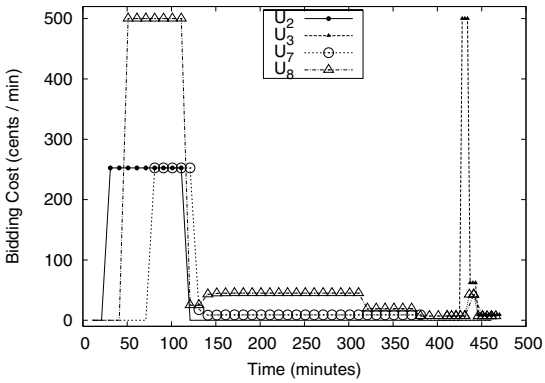

Figure 2. Bidding cost on R_1 of domain A.

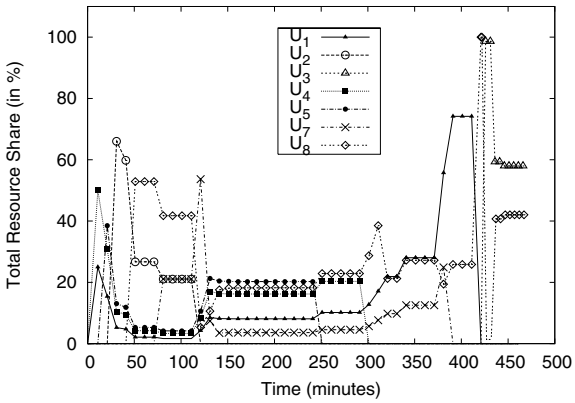

Figure 3. A percentage of total resource share on R_1 of domain A.

5.3. Analysis and Result

Due to space constraints, we only discuss the bidding cost and allocated resource shares of resources R_1 and R_4. In addition, users U_0, U_6 and U_9 are omitted in these figures because their bids are similar to U_1 and U_4.

Figure 2 displays the bidding cost on resource R_1 for users U_2, U_3, U_7, and U_8. They are selected because they vary their bid over time. In Figure 2, user U_8 puts a maximum bid of 500 cents per minute from time 49 to 120 (in simulation minutes). As a result, U_8 gets the most resource share as shown in Figure 3. However, U_8's share drops when U_7 submits tasks at time 77. With a bidding price equal to that of U_2, U_7 gets an equal resource share. This scenario demonstrates how the PS scheduler with an auction model works. A similar pattern is also displayed on R_4 in Figure 5 and 4, from time 0 to 200.

In Figure 5, user U_3's tasks are still executing on resource R_4 at time 400. However, the deadline for U_3 comes

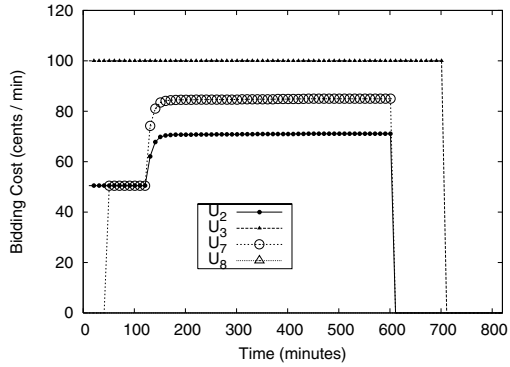

Figure 4. Bidding cost on R_4 of domain B.

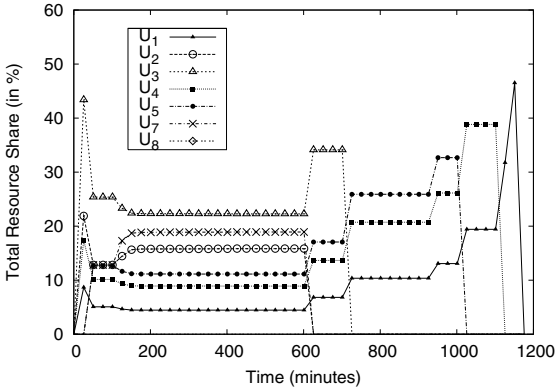

Figure 5. A percentage of total resource share on R_4 of domain B.

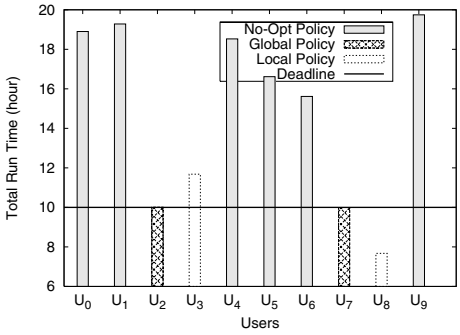

Figure 6. Total runtime for each user (lower is better).

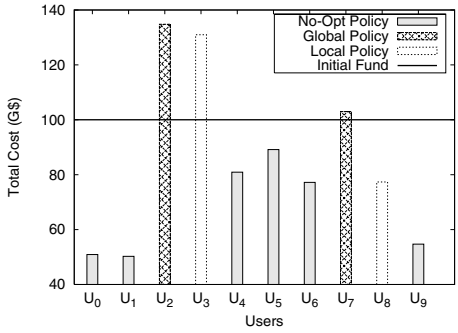

Figure 7. Total cost for each user (lower is better).

at around time 800. With many remaining tasks needing to be completed, U_3 of domain B schedules them to all available local and global resources. Some of these tasks are sent to R_1 of domain A later at time 426, as shown in Figure 3. This scenario demonstrates how a local policy of time optimization algorithm used by U_3 works if a deadline is approaching soon with many remaining tasks.

The impact of user U_3 submitting tasks to resource R_1 makes the resource share of U_8 drops significantly as shown in Figure 3. This is because at time 420, U_3 puts the maximum bid, whereas U_8's bid is near to the minimum bid as depicted in Figure 2. After that, U_8 increases their bid, whereas, U_3 decreases their bid to conserve money since only a few resource shares are needed to complete the tasks before the deadline time. This scenario demonstrates how both local and global policy dynamically adjusts a user bid based on resource share and deadline accordingly.

Figure 6 shows the total run time for each user, with a deadline of 10 hours. As expected, users with a time optimization algorithm (local or global) manage to complete their applications faster than users with No-Opt policy.

Only user U_3, that uses a local policy, is unable to finish before the given deadline. This is because there are many tasks from other users competing on the slowest resource R_4 in VO domain B, as depicted in Figure 5. In contrast, U_8 never submits to R_4 because the user is in VO domain A. Hence, U_8 is able to complete much faster than U_2 and U_7.

Figure 7 shows the total cost for each user. As expected, users with a No-Opt policy have cheaper total cost. Also, user U_8, that uses local policy, has much lower costs than U_2 and U_7 costs that use global policy. This is because all of U_8's tasks are executed on local resources only. As mentioned previously, submitting tasks to global resources will incur a 20% surcharge fee on top of a user bid.

User U_8 also has a cost lower than U_4 and U_5 because U_8 manages to finish much faster than these users. In addition, U_4 and U_5 submit some of their tasks to global resources. Therefore, a combination of high bidding price for a short period of time in a local VO domain costs significantly less than a low bidding price for a longer duration.

User U_3 performs much worse than U_8, although both

users use the same local policy. This is because U_3 puts the maximum bid on resource R_4 throughout the whole period of time, as depicted in Figure 4, which is necessary to complete the tasks on time. In addition, U_3 submits remaining tasks to R_1 in other VO domain that incurs a higher cost, as shown in Figure 3. These two factors contribute to the high total cost of U_3.

6. Conclusion and Further Work

In this paper, we have proposed a time optimization algorithm that aims to minimize the total run time of a user application in order to satisfy a specified deadline. In particular, we consider bag-of-task (BoT) applications, where each application consists of parallel and independent tasks. Moreover, we choose a scenario where users and resources can be part of one or more Virtual Organization (VO) domains. Therefore, we have two variants of this algorithm, one that considers multiple VOs (global policy), and the other one considers only one VO (local policy).

A global policy of the time optimization algorithm submits user tasks to all resources, regardless of their VO domains. Therefore, users with this policy are able to finish their applications within a specified deadline. However, the cost to execute these applications are high, because global resources will charge a surcharge fee.

A local policy of the time optimization algorithm initially submits user tasks to resources that belongs to the same VO as the user. If deadline is approaching soon, it then send remaining tasks to all available resources, including those in other VO domains. The main incentive of this policy is to minimize cost.

The effectiveness of this policy is mixed. If resources of a VO domain have faster computing power, then local users are able to finish their applications much faster and cheaper than users with a global policy, because they do not need to use global resources. In contrast, a slow resource will make local users send their tasks to other resources if deadline is near. Hence, the total cost of execution is similar to users with a global policy. However, this slow resource may cause tasks to be completed longer. As a result, the users may miss their specified deadline.

In the future, we are planning to incorporate task preemption into resource allocation. This allows users to move some tasks on a busy or slow resource to other resources. In addition, we are considering another policy that tries to finish a user application within a fixed budget.

Acknowledgement

We thank Uros Cibej and Wolfram Schiffmann for their discussions on the paper. We also thank Chee Shin Yeo and Hussein Gibbins for their comments.

References

[1] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: an experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 2002.

[2] R. Buyya, D. Abramson, and J. Giddy. Nimrod-G: An architecture for a resource management and scheduling system in a global computational grid. In *Proc. of the 4th Intl. Conference & Exhibition on High Performance Computing in Asia-Pacific Region (HPC Asia'00)*, Beijing, China, May 2000.

[3] R. Buyya and M. Murshed. GridSim: A toolkit for the modeling and simulation of distributed management and scheduling for grid computing. *The Journal of Concurrency and Computation: Practice and Experience*, 14:13–15, 2002.

[4] B. N. Chun and D. E. Culler. REXEC: a decentralized, secure remote execution environment for clusters. In *Proc. of the 4th Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing*, Toulouse, France, January 2000.

[5] W. Cirne, F. Brasileiro, J. Sauve, N. Andrade, D. Paranhos, E. Santos-Neto, and R. Medeiros. Grid computing for bag of tasks applications. In *Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and E-Goverment*, September 2003.

[6] L. B. Costa, L. Feitosa, E. Araujo, G. Mendes, R. Coelho, W. Cirne, and D. Fireman. MyGrid: A complete solution for running bag-of-tasks applications. In *Proc. of the SBRC 2004 – Salao de Ferramentas (22nd Brazilian Symposium on Computer Networks – III Special Tools Session)*, May 2004.

[7] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[8] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *The International Journal of Supercomputer Applications*, 15(3), 2001.

[9] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang, and B. A. Huberman. Tycoon: an Implemention of a Distributed Market-Based Resource Allocation System. Technical Report arXiv:cs.DC/0412038, HP Labs, USA, Dec. 2004.

[10] A. Oram, editor. *Peer-to-peer: Harnessing the Power of Disruptive Technologies*. O'Reilly Press, 2001.

[11] J. Regehr. Some guidelines for proportional share CPU scheduling in general-purpose operating systems. In *Work in progress session of the 22nd IEEE Real-Time Systems Symposium (RTSS 2001)*, London, UK, Dec. 2001.

[12] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. K. Baruah, J. E. Gehrke, and C. G. Plaxton. A proportional share resource allocation algorithm for real-time, time-shared systems. In *IEEE Real-Time Systems Symposium*, December 1996.

[13] Standard Performance Evaluation Corporation http://www.spec.org/.

[14] 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.

[15] L. Winton. Data grids and high energy physics: A melbourne perspective. *Space Science Reviews*, 107(1–2):523–540, 2003.