Scheduling Parallel Applications on Utility Grids: Time and Cost Trade-off Management

Saurabh Kumar Garg [1], Rajkumar Buyya[1] and H. J. Siegel[2]

[1]Grid Computing and Distributed Systems Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010, Australia
Email: {sgarg,raj}@csse.unimelb.edu.au

[2]Electrical and Computer Engineering Department, and
Computer Science Department
Colorado State University
Fort Collins, USA
Email: HJ@ColoState.edu

Abstract

With the growth of Utility Grids and various Grid market infrastructures, the need for efficient and cost effective scheduling algorithms is also increasing rapidly, particularly in the area of meta-scheduling. In these environments,users not only may have conflicting requirements with other users, but also they have to manage the trade-off between time and cost such that their applications can be executed most economically in the minimum time. Thus, choosing of the best Grid resources becomes a challenge in such a competitive market. This paper presents two novel heuristics for scheduling parallel applications on Utility Grids that manage and optimize the trade-off between time and cost constraints. The performance of the heuristics is evaluated through extensive simulations of a real-world environment with real parallel workload models to demonstrate the practicality of our algorithms. We compare our scheduling algorithms against other common algorithms used by current meta-schedulers. The results shows that our algorithms outperform other algorithms by minimizing the time and cost of application execution on Utility Grids.

Keywords: Grid market, scheduling, meta-broker, cost.

1 Introduction

Grid computing enables the harnessing of a wide range of heterogeneous, distributed resources for executing compute- and data-intensive application. Recently, it has been rapidly moving towards a pay-as-you-go model wherein providers expect an economic compensation for the computational resources or services offered to users. Thus, Grid computing has gained a lot of attention from industry leaders such as IBM, HP, Intel and Sun which are involved in this business. For example, IBM has "e-business on demand", HP has "Adaptive enterprise" and Sun Microsystems has "pay as- you-go".

On one side, there are users with applications to execute and, on the other side, there are providers willing to offer their resources or computing services in return for regular payments. Environments with this decoupling of users from providers are generally termed as Utility Grids. Resource providers price their goods to reflect supply and demand in order to make a profit or to regulate consumption. Scheduling in Utility Grids is complex due to the distributed ownership of resources. Moreover, consumers and providers are independent from one another and have different access policies, scheduling strategies and objectives (Chun & Culler 2002). Previous work has proposed Grid market infrastructures (Abramson et al. 2002)(Neumann et al. 2007)(Altmann et al. 2007) for Utility Grids. Although these works provide the basis for resource markets, application scheduling considering aspects such as the distributed resource ownership and cost minimization under these scenarios is still in its infancy. Grid brokers (meta-schedulers) are part of these infrastructures which work on the behalf of users and mediate access to distributed resources by discovering suitable resources for a given user application and optimally mapping jobs to resources. Existing Grid brokers, that consider either cost minimization or time minimization, are generally single-user based (Yu et al. 2005)(Abramson et al. 2002). These single-user brokers may lead to sub-optimal schedules and are certainly not designed with the aim of minimizing the cost and time for a group or community of Grid users. In Utility Grids, users can make a reservation with a service provider in advance to ensure the service availability, and users can also negotiate with service providers on Service Level Agreements for required QoS (Buco et al. 2004).

In this work, we focus on meta-scheduling of different applications from a community of users considering a commodity market. In commodity markets, service providers primarily charge the end user for services that they consume based on the value they derive from it. Pricing policies are based on the demand from the users and the supply of resources is the main driver in the competitive, commodity market models. Therefore, a user competes with other users and a resource owner with other resource owners. The financial institution Morgan Stanley is an example of a user community that has various branches across the world. Each branch has computational needs and QoS constraints that can be satisfied by Grid resources. In this scenario, it is more appealing for the company to schedule various applications in a coordinated manner. Furthermore, another goal is to minimize the cost of using resources to all users across

the community (the company in this case). Therefore, we study the problem of resource scheduling with the goal of minimizing overall execution time and cost. This scheduling problem, which aims to minimize the cost of using resources for all users across the community, is found to be NP-hard due to its combinatorial nature (Martello & Toth 1981). The problem becomes more challenging when a user has to relax its QoS requirements such as makespan under limited budget constraints. The users sometimes may prefer to use cheaper services with a relaxed QoS that is sufficient to meet their requirements. Thus, the user has to choose between multiple conflicting optimization objectives. This research is not only strongly NP-hard, but also non-approximable, i.e., it cannot be approximated in polynomial time within arbitrarily good precision (Kumar et al. 2007). Moreover, the scheduling in Utility Grids needs to be online which further add to the challenge. Hence, we propose heuristics to solve the problem.

In this work, first, we propose two meta-scheduling online heuristics Min-Min Cost Time Tradeoff (MinCTT) and Max-Min Cost Time Tradeoff (MaxCTT) to manage the trade-off between overall execution time and cost and minimize them simultaneously on the basis of a trade-off factor. The trade-off factor indicates the priority of optimizing cost over time. These heuristics can be easily integrated in existing meta-brokers (or meta-schedulers) of Grid Market Infrastructures (Neumann et al. 2007)(Altmann et al. 2007). Second, in order to study the effectiveness and efficiency of the proposed heuristics, we evaluated our heuristics by an extensive simulation study. These heuristics can run in either batch mode or immediate mode(Maheswaran et al. 1999). In the batch mode, the meta-broker waits for a certain time interval (called schedule interval). Then at the end of the schedule interval, the meta-broker allocates all user applications (that are submitted during that interval) to available resources. In contrast, immediate mode heuristics immediately map a task to some machine in the system for execution upon the arrival of the application. In our simulation we have studied the heuristics in batch mode.

The rest of paper is organized as follows. In the next section, we discuss related cost- and time-based scheduling heuristics and meta-schedulers. Section 3 presents the system model and details of our scheduling mechanism are presented in Section 4. Sections 5 presents the experimental setup used for performance evaluation and Section 6 discusses the results. Finally, we conclude the paper and present future work in this direction.

2 Related Works

The research on meta-scheduling mechanisms in Utility Grids can be divided into two parts on the basis of market models, i.e., auctions and commodity market models. As our work is relevant for commodity markets in Grid, in this section we compare our with other resource allocation mechanisms for this market model. Resource management using economy-based principles and market-oriented models have proven to be useful for scheduling applications in Grids (Theliotis et al. 2004).

The work that is most related to this paper is by Buyya et al. (2005), Wolski et al. (2001), Feng et al. (2003) and Dogan & Ozgiiner (2002). In our previous work, Gridbus Broker (Buyya et al. 2005) and Nimrod/G (Abramson et al. 2002), a greedy approach is proposed to schedule a parameter sweep application with deadline and cost constraints. To be precise, our this previous work on Grid scheduling was focused on application-level scheduling, i.e., a personal broker for the efficient deployment of an individual application on Utility Grids. In contrast, this current paper is focused on scheduling of many applications from multiple users having different QoS requirements with the aim of global optimization.

Gcommerce (Wolski et al. 2001) is another economic-based study that applies strategies for pricing Grid resources to facilitate resource trading. It compares auction and commodity market models using these pricing strategies. Feng et al. (2003) proposed a deadline cost optimization model for scheduling one application with dependent tasks. These studies have some limitations: (1) algorithms proposed are not designed to accommodate concurrent users competing for resources; (2) the application model is for an independent task or parametric sweep application. In this work, we have modelled parallel applications submitted by concurrent users. Similarly, Dogan & Ozgiiner (2002) proposed a meta-scheduling algorithm considering many concurrent users, but the application model assumed that each application consists of one task and each application is independent. In this paper, we have considered multiple and concurrent users competing for resources in a meta-scheduling environment to minimize the combined cost and time of all user applications.

Many Genetic Algorithms (GA) based heuristics are also proposed in the literature. Kim & Weissman (2004) proposed a novel GA-based algorithm which schedules a divisible data intensive application. Di Martino & Mililotti (2002) presented a GA-based scheduling algorithm where the goal of super-scheduling was to minimize the release time of jobs. These GA-based heuristic based solutions do not consider QoS constraints of concurrent users such as budget and deadline. Singh et al. (2007) presented a multi-objective GA formulation for provisioning resources for an application using a slot-based resource model to optimize cost and performance. Due to the time consuming nature of GA, these heuristics are not suitable for online meta-scheduling.

For scheduling approaches outside Grid computing, Min-Min, Min-Max and Sufferage (Maheswaran et al. 1999) are three major task-level heuristics employed for resource allocation. As they are developed based on specific domain knowledge, they cannot be applied directly to Grid scheduling problems, and hence have to be enhanced accordingly.

The main contribution of this paper is thus to design two heuristics to manage and optimize the trade-off between cost and execution time of user application in a concurrent user's environment for Utility Grids. We adopt some ideas from Min-Min and Min-Max heuristics to design our algorithm.

3 Meta-Broker System

The meta-broker presented in this work envisions future market models (Neumann et al. 2007) where various service providers with large computing installations and consumers from educational, industrial and research institutions will meet (Figure 1). In this model, service providers sell the CPU time slots on their resources (clusters or supercomputers) and the consumers (or users) will buy these time slots to run their applications. The meta-broker may have control over allocations to some or all processors in a resource for some time intervals. This scenario can be formulated as an economic system with three main participants:

- **Service Providers:** Each of the resources (cluster, servers, supercomputer) can be considered as

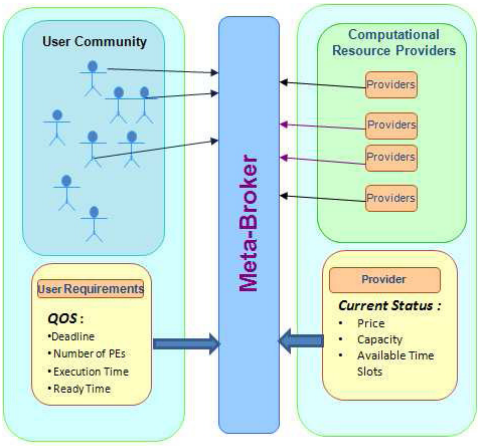


Figure 1: Meta-Broker System

a provider of services such as CPU time slots. Each free CPU slot includes two parameters: number of processors and time for which they are free. Providers have to satisfy requests of the local users at each site and Grid user requests that arrive through the meta-broker. Providers assign CPUs for the exclusive use of the meta-broker through advanced reservation, and supply information about the availability of CPUs and usage cost per second at regular intervals. The economic system considered here is co-operative in nature, that is, the participants trust and benefit each other by co-operating with each other. Therefore, the possibility of providers supplying wrong or malicious information is discounted. It is assumed that service price does not change during the scheduling of applications.

- **Users:** Users submit their applications to the meta-scheduler for execution at the resources in the computing installation/Grid. The users require that the applications be executed in the most economic and efficient manner. The users also can provide a trade-off factor to indicate the importance of cost over execution time, otherwise it will be set by the meta-broker. The trade-off factor can be calculated by user on the basis of urgency and budget for executing the application. In the current system, we assume user applications are based on the parallel application model, that is, the the application requires a certain number of CPUs simultaneously on the same Grid resource for certain time interval.

- **Meta-Broker:** The meta-broker uses the information supplied by the providers and the users to match jobs to the appropriate services. The scheduling of user applications is done in batch mode at the end of a Schedule Interval (SI). At the end of a SI, the meta-broker calculates the best schedule for all user applications after negotiating the time slots with the service providers. The objective of the meta-broker is to schedule all user application such that both total time and cost for applications execution are minimized. The proposed meta-scheduling mechanisms are presented in the next section.

4 Meta-Scheduling Mechanisms

In general, users have two QoS requirements, i.e., the processing time and execution cost for executing their applications on pay-per-use services (Yu et al. 2005). The users normally would like to get the execution done at the lowest possible cost in minimum time. Thus, we introduce trade-off factor which indicates the importance level of cost for users over time. In this section, we present our two meta-scheduling heuristics that aim to manage the trade-off between execution cost and time.

4.1 Mathematical model and Terminologies

We model parallel applications submitted by users to meta-broker. Let $n(t)$ be the number of user applications submitted by users during scheduling interval that ends at time t. Every application i requires p_i CPUs for execution. Let $T(t)$ be the set of applications that meta-broker has to schedule at time t. The estimated time to compute (ETC) values of each application on each compute resource are assumed to be known based on user-supplied information, experimental data, application profiling or benchmarking, or other techniques. The performance estimation for resource services can be achieved by using existing performance estimation techniques (e.g. analytical modelling (Nudd et al. 2000), empirical (Cooper et al. 2004) and historical data (Smith et al. 1998, Jang et al. 2005)) to predict task execution time on every discovered resource service. As a result, the application execution time can be obtained for different resources. The assumption of ETC information is common practice in resource allocation study (Xu et al. 2001). We assume that an application cannot be executed until all of the required CPUs are available simultaneously. Let $m(t)$ be the total number of service providers available and $R(t)$ is the set of service providers available during scheduling interval end at time t. Each service provider has m_i CPUs to rent. Let c_j be the cost of using a CPU on resource j per unit time.

Let $s(i,j)$ and $f(i,j)$ be the submission time and finish time of application i on resource j, respectively. The response time of application i is defined as

$$\alpha(i,j) = f(i,j) - s(i,j)$$

The average execution time of application i is given by

$$\beta_i = \frac{\sum_{j \epsilon R(t)} ETC(i,j)}{m(t)}$$

The cost spent in execution of application i on resource j is given by

$$c(i,j) = c_j \times p_i \times ETC(i,j)$$

The average cost of execution of application i is given by

$$\gamma_i = \frac{\sum_{j \epsilon R(t)} c(i,j)}{m(t)}$$

Thus, given δ is the trade-off factor for all user applications, the trade-off cost metric for each user application is given by,

$$\phi(i,j,t) = \delta \frac{c(i,j)}{\gamma_i} + (1-\delta)\frac{\alpha(i,j)}{\beta_i} \qquad (1)$$

Thus, the objective of our scheduling algorithm is to minimize the summation of trade-off metric for all user applications, i.e.,

$$minimize(\sum_{\forall (i \epsilon T(t), t)} min_{\forall j} \phi(i,j,t))$$

The scheduling problem is to map every appication $i \epsilon T(t)$ onto a suitable resource $j \epsilon R(t)$ to minimize the total execution time and cost of all user applications.

Algorithm 1: Pseudo code for MinCTT

Input: set of applications (submission time, execution time, CPUs required) and resources (time slots, number of available CPUs)
Output: Mapping of applications to resources
1 Collect all user applications untill *Schedule Interval* ends
2 Get list of available time slots for all resources
3 **foreach** *user application* u_i **do**
4 **foreach** *each resource* r_j **do**
5 Find all feasible time slots
6 Find time slot TS which minimizes cost metric $\phi(i,j,t) = \delta \frac{c(i,j)}{\gamma_i} + (1-\delta)\frac{\alpha(i,j)}{\beta_i}$
7 Insert TS and resource pair in feasible schedule queue S
8 **endfch**
9 $(TS_i, r_j) \leftarrow$ element with minimum cost metric value from S
10 Insert $(u_i, (TS_i, r_j))$ pair in a queue K
11 **endfch**
12 $(u, (TS, r)) \leftarrow$ element with minimum cost metric value from K
13 Allocate time slot TS on resource r to user application u
14 Update the time slots list for resource r
15 Remove u from user application list
16 Repeat $3-15$ until all applications are allocated

4.2 Min-Min Cost Time Trade-off (MinCTT) Heuristics

MinCTT is based on the concept of Min-Min heuristic (Maheswaran et al. 1999)(Ibarra & Kim 1977). For each user application, MinCTT finds the time slot on a resource with minimum value of cost metric as defined in (1). From these user application/time slot pairs, the pair that gives the overall minimum is selected and that application scheduled onto that time slot of the resource. This procedure is repeated until all of the user applications have been scheduled. The pseudo code for MinCTT is given in Algorithm 1.

4.3 Max-Min Cost Time Trade-off (MaxCTT) Heuristics

MaxCTT is based on the concept of Max-Min heuristic (Maheswaran et al. 1999)(Ibarra & Kim 1977). This algorithm removes fragmentation from the time slot reservations. For each user application, first MaxCTT finds the time slot on a resource with minimum value of cost metric as defined in (1). Finally, from these user application/time slot pairs, the pair that gives the overall maximum is selected and that application scheduled onto that time slot of that resource. This procedure is repeated until all of the user applications have been scheduled. The pseudo code for MaxCTT is given in Algorithm 2.

Algorithm 2: Psuedo code for MaxCTT

Input: set of applications (submission time, execution time, CPUs required) and resources (time slots, number of available CPUs)
Output: Mapping of appications to resources
1 Collect all user applications untill *Schedule Interval* ends
2 Get list of available time slots for all resources
3 **foreach** *user application* u_i **do**
4 **foreach** *each resource* r_j **do**
5 Find all feasible time slots
6 Find time slot TS which minimizes cost metric $\phi(i,j,t) = \delta \frac{c(i,j)}{\gamma_i} + (1-\delta)\frac{\alpha(i,j)}{\beta_i}$
7 Insert TS and resource pair in feasible schedule queue S
8 **endfch**
9 $(TS_i, r_j) \leftarrow$ element with minimum cost metric value from S
10 Insert $(u_i, (TS_i, r_j))$ pair in a queue K
11 **endfch**
12 $(u, (TS, r)) \leftarrow$ element with maximum cost metric value from K
13 Allocate time slot TS on resource r to user application u
14 Update the time slots list for resource r
15 Remove u from user application list
16 Repeat $3-15$ until all applications are allocated

4.4 Time Complexity

The main operations performed during MinCTT and MaxCTT for a scheduling interval are the following

- To allocate any resource to an application, the number of iteration is to be done over each user application and resource i.e. $m(t)n(t)$ times
- In each iteration (step 5 to 8 in Algorithm 1), time slot with minimum execution time is to be searched. This is of order of available time slots. For resource j, the number of available time slots for an application i at time t is given by $TS(j,i,t)$.
- the above operations are to be done for each application, i.e., $n(t)$ times

Therefore, the resultant complexity of the meta-scheduling mechanism is combination of above operations, i.e., $O(n^2 \sum_{j \epsilon R(t)} TS(j,i,t))$.

5 Simulation Setup

For our experiments, we use GridSim (Buyya & Murshed 2002) to simulate our meta-scheduler model and Grid testbed. The simulation facilitates evaluation as the same testbed environment can be repeated for different approaches. User applications are modeled as parallel applications which require all CPUs to be allocated at the same time and on same resource. About 1,000 user applications are generated according to the Lublin workload model (Lublin & Feitelson 2003). The model specifies the arrival time, number of CPUs required, and executiong time (μ) of application. We divided the arrival times by 1000 to reduce the overall time to run the experiments. Since the generated workload gives execution time on one resource, the ETC matrix is thus generated using random distributions. The variation of the application's execution time on different resources can be high or low. A high variation in execution time of the same application is generated using the gamma distribution method presented by Ali et al. (2000). In the gamma distribution method (Ali et al. 2000), a mean task execution time and coefficient of variation (COV) are used to generate ETC matrices. The mean task execution time of an application is set to μ and a COV value of 0.9 is used. Similarly, the low variation in the execution time is generated using uniform distribution with minimum value of μ and standard deviation of 20 sec.

The computing installation modeled in our simulation is that of a subset of the European Data Grid(EDG) 1 testbed (Hoschek et al. 2000) which contains five Grid resources spread across four countries connected via high capacity network links. The configurations assigned to the resources in the testbed for the simulation are listed in Table 1. The configuration of each resource is decided so that the modeled test bed would reflect the heterogeneity of platforms and capabilities that is normally the characteristic of such installations. All the resources were simulated as clusters of Processing Elements (PEs) or CPUs that

Table 1: Simulated EDG Testbed Resources

Resource name(Country)	Number of PEs	Single PE rating (MIPS)	Inconsistent execution price (G$)	Consistent execution price (G$)
RAL(UK)	20	1320	0.0074	0.0353
Imperial College(UK)	26	1330	0.0100	0.1424
NorduGrid (Norway)	265	1100	0.0139	0.0032
NIKHEF (Netherlands)	54	1166	0.0097	0.0069
Lyon (France)	60	1160	0.0095	0.0061

employed easy backfilling policies and allow advance reservation in order to improve responsiveness. The average initial price of using each PE on a Grid resource is given in Table 1. These resources send the availability of time slots to the meta-broker regularly. The schedule interval of the meta-broker is 50 simulation seconds.

We compare our proposed heuristics (denoted as MinCTT and MaxCTT) with a common heuristic which is used in previous work i.e. cost based Greedy heuristic (Greedy). This approach is derived from the cost optimization algorithm in Nimrod-G (Abramson et al. 2002), which is initially designed for scheduling independent tasks on Grids and thus enhanced for parallel applications. The Greedy heuristic sorts services by the value of cost metrics and assign applications to services with the minimum trade-off cost.

We tested our meta-scheduling heuristics (MinCTT and MaxCTT) by performing a series of experiments that compare our algorithm with the Greedy heuristic. The experiments are conducted for the following two cases:

1. **Case 1:** The trade-off factor is set by the meta-broker.The performance of heuristics is studied in two configurations.
   - High variation in execution time of applications for different resources
   - Low variation in execution time of applications for different resources
2. **Case 2:** The trade-off factor is provided by each user. The performance of heuristics is studied in four configurations.
   - High variation in execution time and inconsistent prices of resources (HIUC)
   - High variation in execution time and consistent prices of resources (HICC)
   - Low variation in execution time and inconsistent prices of resources (LOUC)
   - Low variation in execution time and consistent prices of resources (LOCC)

The consistent prices of resources means as the pricing of resources increases the execution time of applications will decrease. It means that if an application has highest execution time on a resource, the resource will be cheapest one. In other words the price of the slowest resource will be lowest. Otherwise, the pricing of resources will be inconsistent. The prices for resources considered in the CASE I are all inconsistent.

The two metrics used to evaluate the scheduling approaches are overall makespan and average execution cost. The former indicates maximum time when all the submitted applications finish execution, while the latter indicates how much it costs to schedule all the applications on the testbed.

6 Analysis of Results

This section shows the comparison between MaxCTT, MinCTT and Greedy heuristics. This section also shows how the proposed heuristics reduces execution cost and makespan in different scenarios.

6.1 CASE 1: Trade-off Factor Set by Meta-broker

The results for the two types of variation in execution time with varying trade-off factor is shown in Figure 2 and 3. This section presents the effect of different trade-off factors on the performance of heuristics. Both MaxCTT and MinCTT outperformed Greedy heuristic in optimizing the overall execution cost and makespan.

It can be noted from Figure 2(a) and 3(a) that as the trade-off factor is increasing, the overall total execution cost is decreasing. This is because of the increase in scheduling of more applications on the cheaper resources due to increase in the weight of cost over time. The effect of variation in execution time of applications across various resources can be seen clearly from these figures. In Figure 2(a)) the decrease in total execution cost is more in comparison to Figure 3(a). Due to the low variation in execution time, with increase in trade-off factor, execution time of an application doesn't effect the cost metric as defined in (1). Thus, execution cost dominates in this case which results in low execution cost when $trade-offfactor = 1$ in Figure 2(a). while change in overall execution cost in other case remains approximately same i.e., in 3(a).

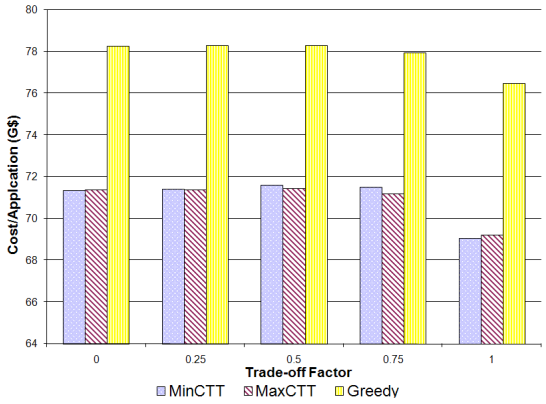
In Figure 2(b), the makespan is increasing with trade-off factor for both Greedy and MinCTT but in the case of MaxCTT the trend is not fixed. This is because in some cases MaxCTT results in the schedule with less fragmentation which results in decrease of makespan. In Figure, 2(b), we can note that there is slight decrease in makespan with increase in value of trade-off factor. This trend is not expected as an increase in the trade-off factor indicates that weight of time in cost metric (1) should decrease and thus results in increase of makespan. This is because of the inconsistent pricing of resources. Thus, with the increase in trade-off factor, many application are executed on resources which are not only cheaper but also faster.

6.2 CASE 2: Trade-off Factor Set by User

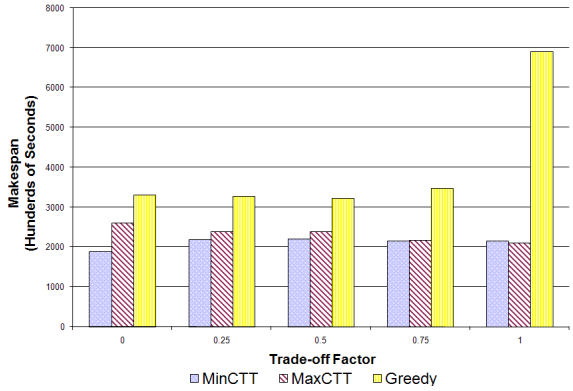
This section discusses the performance of the heuristics in four different configurations of ETC matrix and resource pricing.

6.2.1 Impact on User i.e. Makespan and Execution Cost

In Figure 4, the overall execution cost and makespan of all user applications is compiled for four different configurations. The Greedy heuristic performed the



(a) Overall Average Cost of Execution



(b) Overall Makespan of Applications

Figure 2: Low Execution Time Variation

worst by generating the most expensive schedule with the maximum makespan in almost all four configurations. This is due to the fact that Greedy heuristics does not consider the effect of other applications in the meta-broker while generating the schedule for any application. Moreover, in the case of the LOCC configuration in Figure 4(a), the anomaly in the usual behavior of all heuristics shows how well MinCTT and MaxCTT are managing the time and cost trade-off which results in lower values of makespan with very slight increase in the execution cost as observed from Figure 4(b) for LOCC configuration. Figure 4(a) shows that MinCTT gives the schedule with almost same overall execution cost as MaxCTT , while Figure 4(b) shows that MinCCT gives the schedule with lower makespan than MaxCTT except for the HICC configuration.

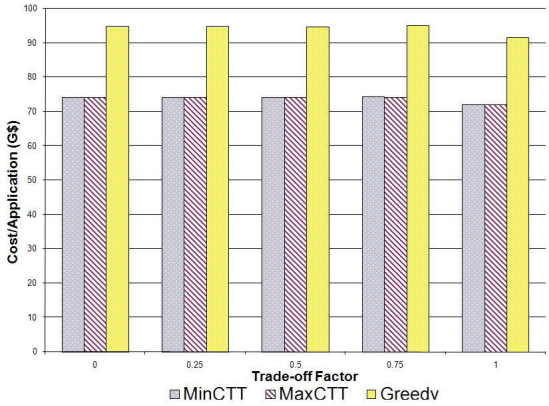
6.2.2 Application Distribution on Resources

Figure 5 shows how the applications are distributed on various resources by the meta-scheduling heuristics in four different configurations. This measure is taken to study how the pricing of resources affects the selection process of the heuristics. In Figure 5(a) and 5(b), it can be observed that, a maximum number of applications are allocated on NorduGrid in all the configurations i.e. LOUC, HIUC, LOCC and HICC. This is due to the fact that NorduGrid has maximum CPUs thus more applications can be scheduled which will result in lower makespan. Moreover, in the case of LOCC and HICC configurations, the price of NorduGrid is the lowest.

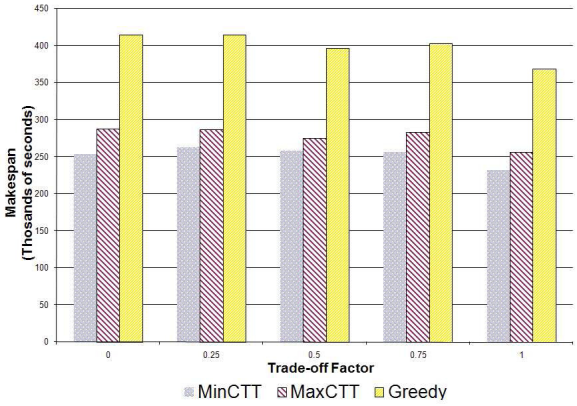
In the case when cost is consistent with the ETC values of application, less variation in execution time of application across resources will results in the assignment of more applications to the cheapest resource as the effect of execution time will be very low. Thus, it can be noted in Figure 5(a)-5(c) that for HICC and LOCC configuration, the number of applications on NorduGrid and LyonGrid, which are the cheapest resources, has increased, while in other cases, they are reduced. For both of LOCC and HICC configurations, we also can observe that on the Imperial College resources more applications are scheduled than RAL which is cheaper. This is due to two reasons, firstly, RAL has only 20 CPUs, thus it can run less number of applications than Imperial College. Secondly, even though Imperial College is expensive, it is the fastest resource thus it will decrease the time factor ($\alpha(i,j)$) in the cost metric as defined in (1).

In Figure 5(a) and 5(b), the reason for lower total execution cost in case of MaxCTT and MinCTT is also clear. MaxCTT and MinCTT has allocated more number of applications on cheaper resources than the Greedy heuristics.

The effect of cost consistency is very low when the variation in execution time of an application is less across different resources. It can be observed in Figure 5(a)-5(c) that the distribution of application in LOUC and LOCC configurations is similar. However, in the case of high variation in execution time of applications across the resources, the effect of cost consistency is quite high. For example, in Figure 5(a), the percentage of applications scheduled by MinCTT on the Imperial College and RAL resources is about 1% in HIUC configuration; which increases to about 20% in case of HICC configuration. A similar pattern



(a) Overall Average Cost of Execution



(b) Overall Makespan of Applications

Figure 3: High Execution Time Variation

of application distribution can be observed in Figure 5(b) and 5(c). The reason for this behavior is due to the trade-off between execution time and cost. For any application, all the heuristics has to choose a resource which is not only run faster but also cheaper. In HIUC configuration, NorduGrid is not only cheapest but also has the maximum number of CPUs. The applications on other resources is allocated to minimize the total makespan.

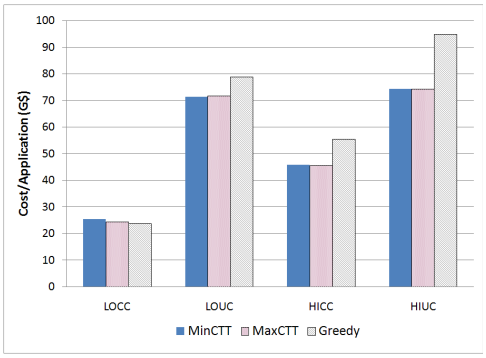
7 Conclusion

Utility Grids provide access to computational services which can be accessed in a secure, transparent and shared market environment on the standard worldwide network. Many users are required to pay for their usage based on their QoS requirements such as makespan and deadline. The concurrent users may generate conflicting schedules to access the same resources which are cheaper and faster. Therefore, many user requirements must be considered during scheduling simultaneously such as cost and execution time. In this paper, we proposed two meta-scheduling heuristics i.e. MaxCTT and MinCTT, that minimize and manage the execution cost and time of user applications. We also have presented a cost metric to manage the trade-off between the execution cost and time.

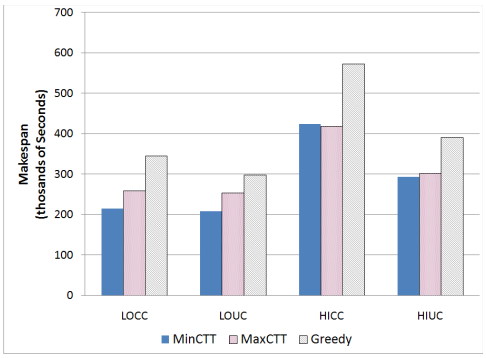
For comparison, we also compared our meta-scheduling heuristics with previously proposed heuristic which is enhanced for the meta-scheduling environment. We evaluated the sensitivity of the proposed heuristics to the changes in the user preferences (the trade-off factor), application execution time and resource pricing. The results show that MaxCTT and MinCTT not only outperforms the Greedy heuristic in optimizing overall execution cost but also in minimizing the overall makespan. In the case when trade-off factor value is chosen by the meta-broker, MinCTT gave the lowest makespan for both the execution time variation (low and high) of applications while MaxCTT gave the lowest execution cost for all trade-off factor (TF) values except for $TF = 1$. In the case when the Trade-off factor is set by users, we studied the behavior of heuristics in four different configurations. Except for LOCC configuration, both MinCTT and MaxCTT generated the cheapest schedule with lowest makespan.

In the future, we would like to enhance our proposed heuristics for the case when resources have different pricing functions based on demand, supply and also usage of resources. We also will develop lower bounds on the highest attainable value of execution cost and makespan. In addition, we will further enhance our heuristics to also support applications with different QoS needs for example, memory and network bandwidth.

Acknowledgements
We would like to thank our colleagues - Marco Netto, Marcos Dias de Assuncao and Chee Shin Yeo - for their comments and suggestions on this paper and on simulator implementation. This research is funded by the International Science Linkage grant provided by the Department of Innovation, Industry, Science and Research (DIISR), Australia Research Council (ARC),the USA National Science Foundation (NSF) under Grant CNS- 0615170, and the Colorado State



(a) Total Execution Cost of Applications



(b) Total Makespan of Applications

Figure 4: Different ETC and Resource Pricing Configurations

University George T. Abell Endowment.

References

Abramson, D., Buyya, R. & Giddy, J. (2002), 'A computational economy for grid computing and its implementation in the Nimrod-G resource broker', *Future Generation Computer Systems* **18**(8), 1061–1074.

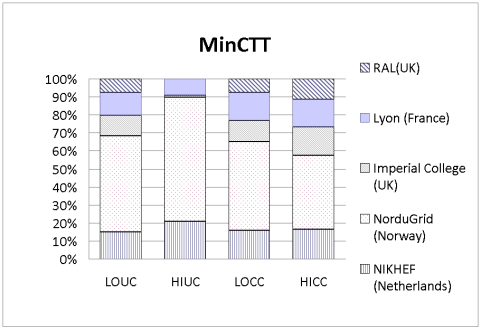
Ali, S., Siegel, H., Maheswaran, M., Hensgen, D. & Ali, S. (2000), 'Representing Task and Machine Heterogeneities for Heterogeneous Computing Systems', *Tamkang Journal of Science and Engineering* **3**(3), 195–208.

Altmann, J., Courcoubetis, C., Darlington, J. & Cohen, J. (2007), GridEcon-The Economic-Enhanced Next-Generation Internet, *in* 'Processdings of the 4th International Workshop on Grid Economics and Business Models, Rennes, France'.

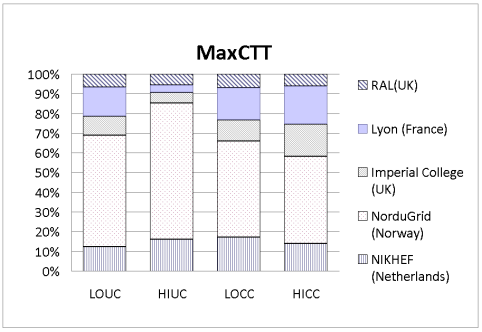
Buco, M., Chang, R., Luan, L., Ward, C., Wolf, J. & Yu, P. (2004), 'Utility computing SLA management based upon business objectives', *IBM Systems Journal* **43**(1), 159–178.

Buyya, R. & Murshed, M. (2002), 'GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing', *Concurrency and Computation: Practice and Experience* **14**(13-15), 1175–1220.

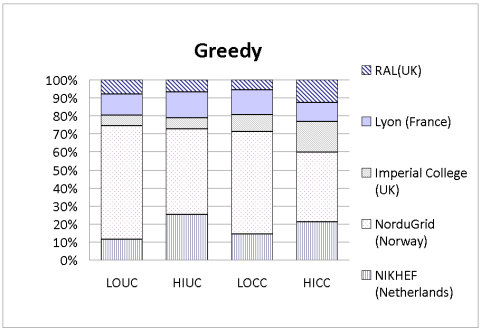
Buyya, R., Murshed, M., Abramson, D. & Venugopal, S. (2005), 'Scheduling parameter sweep applications on global Grids: a deadline and budget constrained cost-time optimization algorithm', *Software Practice and Experience* **35**(5), 491–512.



(a) MinCTT Heuristic



(b) MaxCTT Heuristic



(c) Greedy Heuristic

Figure 5: User Application Distribution on Resources in Different Configurations

Cheliotis, G., Kenyon, C., Buyya, R. & Melbourne, A. (2004), 'Grid Economics: 10 Lessons from Finance', *Peer-to-Peer Computing: Evolution of a Disruptive Technology, Ramesh Subramanian and Brian Goodman (editors), Idea Group Publisher, Hershey, PA, USA* .

Chun, B. & Culler, D. (2002), User-centric Performance Analysis of Market-based Cluster Batch Schedulers, *in* 'Proceedings of 2nd IEEE International Symposium on Cluster Computing and the Grid, Berlin, Germany'.

Cooper, K., Dasgupta, A., Kennedy, K., Koelbel, C., Mandal, A., Marin, G., Mazina, M., Mellor-Crummey, J., Berman, F., Casanova, H. et al. (2004), New grid scheduling and rescheduling methods in the GrADS project, *in* 'Proceedings of 18th International Parallel and Distributed Processing Symposium, New Mexico, USA'.

Di Martino, V. & Mililotti, M. (2002), Scheduling in a grid computing environment using genetic algorithms, *in* 'Proceedings of 16th International Parallel and Distributed Processing Symposium, Florida, USA'.

Dogan, A. & Ozgiiner, F. (2002), Scheduling Independent Tasks with QoS Requirements in Grid Computing with Time-Varying Resource Prices, *in* 'Proceedings of the Third International Workshop on Grid Computing, Maryland, USA'.

Feng, H. et al. (2003), A deadline and budget constrained cost-time optimization algorithm for scheduling dependent tasks in grid computing , *in* 'Proceedings of the Second International Workshop on Grid and Cooperative Computing, Shanghai, China'.

Hoschek, W., Jaen-Martinez, J., Samar, A., Stockinger, H. & Stockinger, K. (2000), Data Management in an International Data Grid Project, *in* 'Proceedings of 1st International Workshop on Grid Computing, Bangalore, India'.

Ibarra, O. & Kim, C. (1977), 'Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors', *Journal of the ACM (JACM)* **24**(2), 280–289.

Jang, S., Taylor, V., Wu, X., Prajugo, M., Deelman, E., Mehta, G. & Vahi, K. (2005), Performance Prediction-based versus Load-based Site Selection: Quantifying the Difference, *in* 'Proceedings of the 18th International Conference on Parallel and Distributed Computing Systems, Las Vegas, Nevada'.

Kim, S. & Weissman, J. (2004), A Genetic Algorithm Based Approach for Scheduling Decomposable Data Grid Applications, *in* 'Proceedings of the 2004 International Conference on Parallel Processing, Las Vegas, NV, USA'.

Kumar, S., Dutta, K. & Mookerjee, V. (2007), 'Maximizing business value by optimal assignment of jobs to resources in grid computing', *European Journal of Operational Research (in Press), http://dx.doi.org/10.1016/j.ejor.2007.12.024* .

Lublin, U. & Feitelson, D. (2003), 'The workload on parallel supercomputers: modeling the characteristics of rigid jobs', *Journal of Parallel and Distributed Computing* **63**(11), 1105–1122.

Maheswaran, M., Ali, S., Siegel, H., Hensgen, D. & Freund, R. (1999), 'Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems', *Journal of Parallel and Distributed Computing* **59**(2), 107–131.

Martello, S. & Toth, P. (1981), 'An algorithm for the generalized assignment problem', *Operational Research* **81**, 589–603.

Neumann, D., Stoesser, J., Anandasivam, A. & Borissov, N. (2007), SORMA-Building an Open Grid Market for Grid Resource Allocation, *in* 'Processdings of the 4th International Workshop on Grid Economics and Business Models, Rennes, France'.

Nudd, G., Kerbyson, D., Papaefstathiou, E., Perry, S., Harper, J. & Wilcox, D. (2000), 'Pace–A Toolset for the Performance Prediction of Parallel and Distributed Systems', *International Journal of High Performance Computing Applications* **14**(3), 228–251.

Singh, G., Kesselman, C. & Deelman, E. (2007), A provisioning model and its comparison with best-effort for performance-cost optimization in grids, *in* 'Proceedings of the 16th International Symposium on High Performance Distributed Computing, California, USA'.

Smith, W., Foster, I. & Taylor, V. (1998), Predicting Application Run Times Using Historical Information, *in* 'Proceedings of IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, Florida, USA'.

Wolski, R., Plank, J., Brevik, J. & Bryan, T. (2001), G-commerce: Market formulations controlling resource allocation on the computational grid, *in* 'Proceedings of the 15th IEEE International Parallel and Distributed Processing Symposium, San Francisco, USA'.

Xu, D., Nahrstedt, K. & Wichadakul, D. (2001), 'QoS and Contention-Aware Multi-Resource Reservation', *Cluster Computing* **4**(2), 95–107.

Yu, J., Buyya, R. & Tham, C. (2005), Cost-Based Scheduling of Scientific Workflow Application on Utility Grids, *in* 'Proceedings of the First International Conference on e-Science and Grid Computing, Washington, DC, USA'.