STATE INDEPENDENT RESOURCE MANAGEMENT FOR DISTRIBUTED GRIDS

Aysan Rasooli, Douglas G. Down
Department of Computing and Software, McMaster University, Main St. West, Hamilton, Canada
{*rasooa, downd*}*@mcmaster.ca*

Keywords: Grid Scheduling; Scheduling Algorithms; Shadow Routing Approach.

Abstract: In practice, a scheduling algorithm should consider multiple objectives. Typically, there are two kinds of objectives. The first is the performance of the system in terms of quantities related to the completion time of tasks, the second is the amount of state information required, which is often measured in terms of quantities such as communications costs. These two objectives are often in tension with one another. In this work, we introduce a scheduling algorithm which simultaneously addresses the objectives listed above namely, minimizing completion times, while requiring *zero* dynamic state information. Using simulation, we show the promising performance of our algorithm, and its robustness with respect to errors in parameter estimates.

1 INTRODUCTION

Task scheduling is an integral part of a distributed computing system. The scheduling algorithm involves matching of application needs with resource availability. Grid scheduling is a problem with multiple objectives, and it is extremely difficult to satisfy all of the objectives with one scheduling algorithm. To the best of our knowledge there is no single Grid scheduling algorithm which is the optimum over all Grid systems with their different applications and features. In this work, we address the scheduling problem of Grid systems whose resources are widely distributed, and there is a considerable communication cost between the resources. The most well-known application of these Grid systems is in the Enabling Grids for E-sciencE (EGEE) (Erwin and Jones, 2009) project. The main contributions of our work are:

- We bring a theoretical idea from the literature (the so-called Shadow Routing algorithm), to a practical scheduling algorithm implemented in Grid systems.
- We modify this basic theoretical approach to be efficient for Grid systems, and study the advantages of the proposed algorithm in widely distributed Grid systems.

In general, the Grid scheduling algorithm should improve the performance of the Grid system, which can be evaluated by various criteria such as Flowtime or Makespan. Furthermore, if an algorithm reduces the amount of state information required at the time of scheduling, this leads to reductions in the communication cost and synchronization overhead. In fact, a large system that requires full state information for scheduling may suffer from increasing limitations due to significant communication and synchronization overheads.

The Shadow Routing method is a robust, generic scheme, introduced in (Stolyar and Tezcan, 2009) for routing of arriving tasks in systems of parallel queues with flexible, many-server pools. This algorithm has proven to achieve good performance levels in queuing systems. However, its structure is designed in a way which is not directly applicable in Grid systems. In this paper we modify the structure of the Shadow Routing approach in two key ways, and introduce a scheduling algorithm for Grid systems, called the Grid Shadow Routing algorithm. First, we change the structure of the basic Shadow Routing algorithm to be applicable for a typical Grid workload model (Iosup et al., 2006). Second, we add to the basic Shadow Routing algorithm, which causes significant improvement in its performance in Grid systems.

The Grid Shadow Routing algorithm uses virtual queues to keep track of the loads on resources in the system. These are used as estimates of the actual queue lengths in the system, thus removing the need to gather real-time load information. The only information that the Grid Shadow Routing algorithm requires are estimates of the lengths of tasks and the execution rates of resources in the system. By using

this information, the algorithm properly balances the loads on the resources. The two estimated parameters are used in most Grid scheduling algorithms. In order to provide this information various prediction methods are applied to forecast them, and in turn to guide task scheduling and load balancing strategies to achieve high performance and more efficient resource usage (Zhang et al., 2006; Lu et al., 2004). An important advantage of our algorithm is that it does not require highly accurate estimates- even highly inaccurate estimates can be satisfactory for our algorithm.

Generally, based on the information that can be used, and the timing of the scheduling decision, scheduling algorithms are classified as either static or dynamic algorithms. Static scheduling algorithms do not use any dynamic state information, but there can be a huge performance degradation in comparison to dynamic algorithms. On the other hand, dynamic scheduling algorithms can make better scheduling decisions, while increasing the communication cost. If a Grid system has low communication overhead, a dynamic scheduling algorithm with full state information can make a significant improvement in the performance of the system. On the other hand, in a widely distributed system in which the time to gather full state information is significant (as in the systems in which we are interested), a dynamic scheduling algorithm which requires full state information can potentially create severe additional overheads. The main advantage of our proposed Grid Shadow Routing algorithm is that it requires *zero* state information from resources at the time of scheduling, and it can achieve much better performance than dynamic algorithms that require full state information. This is particularly advantageous for large, highly loaded systems with widely distributed resources, where communications costs are significant.

Using simulation, we evaluate the Grid Shadow Routing algorithm by comparing to two other algorithms. Minimum Completion Time (MCT) is a dynamic scheduling algorithm, commonly used in Grid systems. This algorithm greedily attempts to reduce the mean completion time, and does not consider the communication overhead in scheduling. The second algorithm that we employ is Join the Shortest Queue, which uses partial state information and does not require task lengths in making scheduling decisions. In order to evaluate the performance of our algorithm in a real system in which the parameters may be estimated inaccurately, we implement our algorithm in a system which has various levels of error in the estimates. Moreover, we evaluate our algorithm by using inaccurate parameters, to show that the significant performance gain of our algorithm can be achieved even in an environment in which there are errors in estimating required parameters.

2 CURRENT ALGORITHM AND RELATED WORK

In this section, rather than presenting a complete survey of current Grid scheduling algorithms, we list two of the commonly used scheduling algorithms for Grid environments, those for which we compare performance with our proposed algorithm.

- *Minimum Completion Time (MCT)*: assigns each task to the resource which has the minimum expected completion time for that task (Dong and Akl, 2006). The expected completion time for a newly arriving task will be computed at each resource; the scheduler collects this state information from all resources and chooses the resource with the minimum expected completion time for execution of the new task. This algorithm can cause a significant improvement in maximum completion time of all tasks. However, it has the cost of requiring full state information, and consequently may have a large communication cost. This algorithm requires the estimated length of incoming task, current resource execution rate, current available bandwidth, and the real-time load on each resource.
- *Join the Shortest Queue* (JSQ*)*: This algorithm assigns each task to the resource which has the smallest number of waiting tasks in its queue. The advantage of this approach is that it does not require the length of tasks to make a scheduling decision. This algorithm just requires one parameter: the real-time number of tasks in each resource queue. However, this parameter should be collected from all resources at the time of scheduling.

3 PROPOSED ALGORITHM

In this Section we describe the problem of task scheduling and our workload model. Then, we provide the details of our proposed scheduling algorithm.

3.1 Workload and System Model

In order to describe and evaluate our scheduling algorithm, we define a workload model based on a typical Grid workload (Iosup et al., 2006). In our workload model, we let the number of resources in the system

be M. The actual resource execution rate for resource r is given by μ_r, and the task length for task k is given by L_k. We assume the use of one of the available estimation methods to provide estimates of resource execution rates for all resources and the length of each incoming task. We define the estimated length of task k as $\hat{L}_k$, and the estimated execution rate of resource r as $\hat{\mu}_r$.

In order to model a widely distributed Grid system, we assume that the Grid network has associated delays. The delay in the network is calculated based on the bandwidth and the load on the Grid network. When a task arrives to the system, the Grid scheduling algorithm is used to route the arriving task to one of the available resources in the system. Here we assume that all local schedulers are using the classical FIFO algorithm, however in general each resource can use its own local scheduling algorithm.

3.2 Grid Shadow Routing Algorithm

The basic Shadow Routing algorithm was first introduced in (Stolyar and Tezcan, 2009) as an algorithm for routing in systems of parallel queues. This algorithm has significant advantage in properly balancing the load of the system without requiring any state information. However, the basic Shadow Routing algorithm is defined on a workload model which is not applicable in Grid systems.

In this Section, we introduce a new Grid scheduling algorithm, called the *Grid Shadow Routing algorithm (Grid Shadow)*, which is presented in Figure 1. In order to apply the idea of the Shadow Routing algorithm in Grid systems, first we should eliminate the class-based setting of the basic Shadow Routing algorithm. As we know, a Grid is a dynamic system in which the resources may join and leave at any time, and various types of tasks may be assigned to the system. So, it is unrealistic to assume that we have predefined types (class) of tasks, and that the execution rate of each class on each resource is known.

Instead of using the workload model of the basic Shadow Routing algorithm, we consider each task separately. We introduce our algorithm based on our typical Grid workload model. As mentioned before, various estimation methods have been introduced to estimate task lengths and resource execution rates (see (Zhang et al., 2006; Lu et al., 2004) for example). Rather than going into detail on any particular estimation method, we simply assume that such estimates have been provided, with associated errors. By applying these two parameters, we estimate the expected execution time of task k on resource r by $\frac{\hat{L}_k}{\hat{\mu}_r}$, where $\hat{L}_k$ and $\hat{\mu}_r$ are estimates of the length of task k and the execution rate of resource r, respectively. Also, we assume that each resource has a virtual queue (Q_r), which is used to estimate loads on resources, and the parameter $\eta > 0$ is used to control the tradeoff between responsiveness of the algorithm and its accuracy. In Grid scheduling, taking into account

- For all resources in the system, set the initial values of Q_r to $\eta Q_r = 1/M$ for all r.
- Upon the arrival of task k, the algorithm performs the following steps :
  1. Among all the resources available in the system (i.e. those with $\hat{\mu}_r > 0$), choose the resource m, such that :
     $$m \in \arg\min_r \{(Q_r + \tfrac{\hat{L}_k}{\hat{\mu}_r}) \times \tfrac{\hat{L}_k}{\hat{\mu}_r}\}$$
     and submit the arriving task to resource m.
  2. The virtual queue of resource m is updated to :
     $$Q_m = Q_m + \tfrac{\hat{L}_k}{\hat{\mu}_m}$$
  3. If condition
     $$\eta \textstyle\sum_r Q_r \geq 1$$
     holds, the following update is performed:
     for each resource r:
     If $(Q_r - c_k > 0)$ then
     $$Q_r = Q_r - c_k$$
     else
     $$Q_r = 0$$
     where $c_k = 2 \times (\max\{\tfrac{\hat{L}_k}{\hat{\mu}_r} \mid \mu_r > 0\})$.

Figure 1: The Grid Shadow Routing Algorithm

the load that an incoming task adds to each resource, is important when the resources are heterogeneous, and the load of the system is moderate or light. The basic Shadow Routing algorithm only considers the current load of each resource in making the scheduling decision for each incoming task. In our scheduling decision, instead of comparing the current loads, we consider the current size of the virtual queue plus the expected load of the incoming task on the corresponding resource. Another way to look at this is that from an analytic perspective, if the load on the system approaches 1 (as in (Stolyar and Tezcan, 2009)), then the effect of the incoming task is negligible. This may not be true in practice, and should be accounted for.

For each incoming task, we aim to increase the total amount in the virtual queues by the minimum possible amount, then the normalization step will be triggered less frequently. This results in less overhead due to scheduling decisions, which improves the performance of the basic Shadow Routing algorithm sig-

nificantly. Science Grid systems have a large number of resources, and the basic Shadow Routing algorithm requires searching over all resources to update all the virtual queues. The algorithm will be more efficient, if it reduces the number of time it triggers the virtual queue updating process.

When a new task arrives to the system, our scheduling algorithm considers three factors for choosing a resource: the current load on each resource, the estimated execution time of the arriving task on each resource, and the effect of the incoming task on each resource load. So, we define the first step of our algorithm as follows: the algorithm compares the quantity $((Q_r + \frac{\hat{L}_k}{\hat{\mu}_r}) \times \frac{\hat{L}_k}{\hat{\mu}_r})$ for all resources, and chooses the resource with the smallest value. The algorithm has a trade-off between a resource which finishes the currently assigned tasks earlier, and a resource which executes the incoming task faster. Also, it aims to minimize the load which is going to be added to each resource. The algorithm adds the estimated execution time of the task on the selected resource to the virtual queue of that resource, which is given by $\frac{\hat{L}_k}{\hat{\mu}_m}$ for task k on resource m.

If the loads on faster resources increase such that the proper balancing of loads is going to be violated, the total virtual queue length of all resources will reach a predefined limit. In this case, the virtual queue lengths of all resources are reduced by a specific amount. This is a normalization step of the algorithm, by which the algorithm is making the virtual queue length of slower resources smaller, and is increasing the chance of slower resources being chosen for executing future tasks. The parameter η should be chosen based on the features of the system. For the workloads considered in this work, we conclude that a good value of η is $1/300$.

Since the Grid Shadow Routing algorithm makes each decision based on the values of virtual queues, if task input rates, or resource execution rates change in the system, no explicit detection of such an event (or any other input rate measurement/estimation) is necessary. The virtual queues automatically readjust and the algorithm starts routing along the new best matchings of resources to tasks.

4 EXPERIMENTAL SET-UP

We use simulation to evaluate the scheduling algorithms. This section gives details of the simulation toolkit used, the performance metrics applied, and the experimental set-up. Simulation models were implemented with the Java package GridSim (Buyya and Murshed, 2002). Depending on the Grid scenario and applications run in the system, there exist different performance metrics for evaluating Grid scheduling algorithms. We use two of the most important performance metrics to evaluate the algorithms from different aspects. The metrics that we consider are: Makespan (the maximum completion time of all tasks), and Flowtime (the average completion time of all tasks).

We consider a Grid system consisting of 50 dedicated resources with different CPU speeds, working in parallel with an overall high load. To simulate a widely distributed Grid system, and because the bandwidth between elements of the system which are far from each other is low, we set the bandwidth inside the elements of the system to be 1 Gbps, and the bandwidth between the scheduler and each of the 50 resources to be 10 Mbps.

As mentioned before, our proposed algorithm is mostly advantageous for EGEE Grids, so we evaluate our algorithm in a real workload from the CERN Grid project. We use a workload from the Grid Workload Archive, in Grid Workloads Format (GWF). This workload is collected from the LCG project. The LCG testbed represents the Large Hadron Collider (LHC) Computing Grid. We use the LCG trace, version 0.1 which is provided by the Grid Workloads Archive (Iosup et al., 2006). We use the first $20,000$ tasks in this trace for our experiment. We run our simulation until $20,000$ tasks arrive to the system and then wait until the system becomes empty.

Our algorithm uses estimates of the task lengths and resource execution rates. However, various estimation methods may have different levels of accuracy. So, we evaluate our algorithm in a system that has various levels of error in the estimated task lengths and resource execution rates. In order to completely study the robustness of our algorithm, we examine cases that have 0% to 40% error in our estimates; however, typically these errors are on the order of 10% (Akioka and Muraoka, 2004). We evaluate our proposed algorithm by considering the error model discussed in (Iosup et al., 2008) for estimating task lengths and resource execution rates. Generally the two models of error in these estimates are:

- *Over and Under Estimation Error.* In our simulations, $\hat{L}_k$ and $\hat{\mu}_r$ are obtained using the following relations: $\hat{L}_k = L_k \times (1 + E_k)$ and $\hat{\mu}_r = \mu_r \times (1 + E_r)$. Here, E_k and E_r are the errors for task lengths and resource execution rates, respectively, which are sampled from the uniform distribution $[-I, +I]$, and I is the maximum error.
- *Over Estimation Error.* The main error models are obtained using the relations $\hat{L}_k = L_k \times (1 + E'_k)$

and $\hat{\mu}_r = \mu_r \times (1 + E'_r)$. The parameters E'_k and E'_r are the errors for task lengths and resource execution rates, respectively, and are sampled from the uniform distribution $[0, +I]$, in which I is the maximum error. This model is used for systems which always over estimate the parameters (powers of resources are estimated to be the maximum amount without considering fluctuation of their power caused by increasing the load).

5 EXPERIMENTAL RESULTS

In this Section we consider the various error models. The complete results of our experiments are provided in (Rasooli and Down, 2011), and for the sake of space, we will not present those results in this paper.

5.1 Over and Under Estimation

In this part, we assume the over and under estimation error model applies. First, we evaluate our algorithm in an environment with accurate resource execution rates, and errors in estimating task lengths, whose result, from the Flowtime perspective, is provided in Figure 2. Then, we consider an environment with

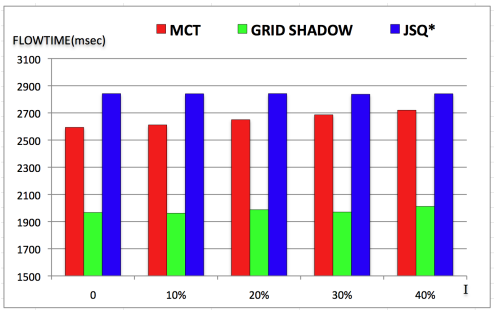

Figure 2: Flowtime-over & under estimating task length

errors in estimating both task lengths and resource execution rates, whose result is provided in Figure 3. Different error levels are considered in these figures. Among the algorithms presented in this work, the MCT algorithm is the only one that uses full state information in order to make scheduling decisions. So, we expect that in the absence of overhead, this algorithm should achieve the smallest Makespan and Flowtime, and should lead to a good balance between the loads of resources. Since our simulations consider a highly loaded system, in which gathering full state information causes large overhead, the MCT algorithm can not achieve good Flowtime compared to our proposed algorithm. However, the MCT algorithm does achieve the best Makespan by minimizing the completion time for each individual incoming task. Generally, minimizing Flowtime can be at the expense of the largest task taking a long time, whereas minimizing Makespan asks that no task takes too long, at the expense of most tasks generally taking a longer time. In summary, minimizing Makespan can result in maximizing Flowtime. So, the poor Flowtime performance of the MCT algorithm results from the combination of its high overhead and its greedy approach in minimizing the completion time for each individual task. Since the Grid Shadow Routing algo-

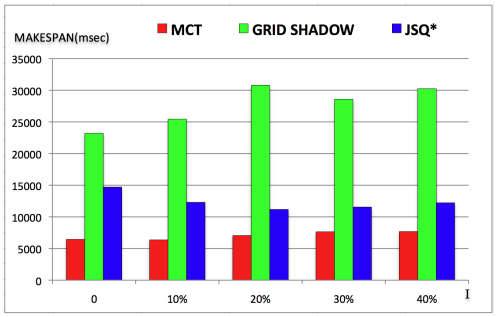

Figure 3: Makespan-over & under estimating task length & resource rate

rithm does not use any state information, one might expect a large difference between its performance and the MCT algorithm. However, the results illustrate that our algorithm has much better Flowtime than the MCT algorithm. This is due to the fact that the Grid Shadow Routing algorithm does not use the greedy view point of the MCT algorithm in optimizing the completion time for any single task. Instead, our proposed algorithm considers a long term approach for minimizing the completion times, and balancing the loads in the system, so it can achieve good performance for aggregate metrics like Flowtime. Another reason for better Flowtime of our algorithm compared to the MCT algorithm is the large overhead of the MCT algorithm in gathering full state information from the system, while our proposed algorithm has no such overhead. As the Grid Shadow Routing algorithm does not have the goal of minimizing the completion time for each individual task, and it considers overall balancing of loads, this can increase the completion time for a small number of tasks, which results in larger Makespans for the Grid Shadow Routing algorithm. Still, its Makespan is competitive with the JSQ* algorithm. We believe that in some Grid systems the Flowtime (which is interpreted as QoS (Maheswaran et al., 1999)) of the system is more important than the Makespan (which is interpreted as throughput of the system). According to the results, even in

systems which have large estimation errors, our proposed Grid Shadow Routing algorithm still has much better Flowtime than the MCT algorithm.

5.2 Over Estimation

In this part, we assume the over estimation error model applies. First, we evaluate our algorithm in an environment with various error levels in estimating task length, whose result is provided in Figure 4.

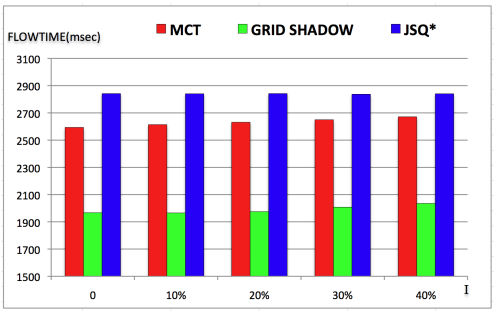

Figure 4: Flowtime-over estimating task length

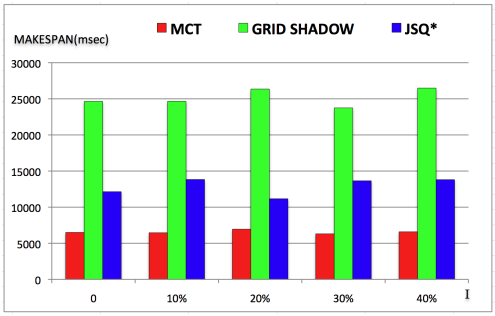

Figure 5: Makespan-over estimating task length and resource rate

Later, we consider an environment with error in both task length and resource execution rate estimation, whose result is provided in Figure 5. Different error levels are considered in these figures. Based on the results, up to 40 percent over estimation does not significantly affect performance of our algorithm. The Grid Shadow Routing algorithm has the best Flowtime and the MCT algorithm has the best Makespan. To summarize the observations in this section, in a real Grid workload, the Grid Shadow Routing algorithm has much better Flowtime than the MCT algorithm, and the Grid Shadow Routing algorithm achieves this performance without collecting any state information. The MCT algorithm achieves the best Makespan at the cost of collecting full state information. So, our Grid Shadow Routing algorithm is a promising candidate for widely distributed, and highly loaded Grid systems.

ACKNOWLEDGEMENTS

This work was supported by the Natural Sciences and Engineering Research Council of Canada. The LCG Grid traces are provided by the HEP e-Science group at Imperial College London.

REFERENCES

Akioka, S. and Muraoka, Y. (2004). Extended forecast of cpu and network load on computational grid. In *Proceedings of the 4th IEEE International Symposium on Cluster Computing and the Grid(CCGrid'04)*, pages 765–772, USA. IEEE Computer Society.

Buyya, R. and 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.

Dong, F. and Akl, S. G. (2006). Scheduling algorithms for grid computing: State of the art and open problems. Technical Report 504, School of Computing, Queens University, Canada.

Erwin, L. and Jones, B. (2009). Enabling grids for e-science: The egee project, egee-pub- 2009-001.

Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., and et al. (2006). The Grid Workloads Archive.

Iosup, A., Sonmez, O., Anoep, S., and Epema, D. (2008). The performance of bags-oftasks in large-scale distributed systems. In *Proceedings of the 17th International Symposium on High Performance Distributed Computing*, pages 97–108.

Lu, D., Sheng, H., and Dinda, P. (2004). Size-based scheduling policies with inaccurate scheduling information. In *Proceedings of the 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, pages 31–38, USA. IEEE Computer Society.

Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., and Freund, R. F. (1999). Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *Proceedings of the 8th Heterogeneous Computing Workshop*, page 30, USA. IEEE Computer Society.

Rasooli, A. and Down, D. G. (2011). State independent resource management for distributed grids. Technical Report CAS-11-01-DD, Department of Computing and Software, McMaster University, Canada.

Stolyar, A. L. and Tezcan, T. (2009). Control of systems with flexible multi-server pools: a shadow routing approach. Bell Labs Technical Memo, revised.

Zhang, Y., Sun, W., and Inoguchi, Y. (2006). Predicting running time of grid tasks based on cpu load predictions. In *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing (Grid06)*, pages 286–292, USA. IEEE Computer Society.