An Adaptive Quality of Service Based Scheduling Algorithm for Wide Area Large Scale Problems[*]

Wilson Lozano and Wilson Rivera
Parallel and Distributed Computing Laboratory
Electrical and Computer Engineering Department
University of Puerto Rico at Mayagüez, Puerto Rico 00680, USA
{wilson.lozano, wilson.rivera}@ece.uprm.edu

Abstract

This paper explores the problem of dynamically scheduling large scale applications over wide area networks and then proposes an adaptive scheduling algorithm to provide quality of service. Our adaptive algorithm takes into account unexpected events and priority fluctuations and gives high priority to jobs with low probability of failure. Experimental results show that the new approach outperforms traditional scheduling approaches in grid computing environments.

1. Introduction

The availability of powerful computers and high-speed network technologies as low-cost commodity components has changed the way we solve large scale problems. These technology opportunities have led to the possibility of using geographically distributed computers as a single, unified computing resource. Grid computing enables coordination, storage and networking of resources across geographically dispersed organizations in a transparent way for users. The first generation of grid technologies has demonstrated the feasibility of grids for addressing challenging large scale problems (e.g. GridPhyN and Teragrid among many others). Next generation of grid applications will be increasingly dynamic. This implies that the current static infrastructures will not be adequate unless adaptive functionalities are provided.

Emerging grid applications demand efficient data and resource management mechanisms. In this paper, we explore the problem of dynamically scheduling large scale applications over wide area networks and then propose an adaptive scheduling framework to provide quality of service. Our adaptive algorithm takes into account unexpected events and priority fluctuations and gives high priority to jobs with low probability of failure. The organization of this paper is as follows. Section 2 discusses the state-of-the-art in wide area scheduling Section 3 describes the proposed wide area scheduling model and the dynamic distributed scheduling algorithm as well as a dynamic framework to implement the scheduling strategy. Experimental results are presented in section 4.

2. State-of-the-Art in Wide Area Scheduling

Although scheduling has been studied in various contexts, with the emergency of grid computing, unique challenges have arisen. Grids are shared infrastructures with no central control, where the applications compete for the best quality of service from remote resources. In addition, grids exhibit fluctuations in the availability of resources and communication latencies over multiple resource administrative domains. The emerging grid technology [1, 2] has led to the need of a new generation of applications capable of adapting its execution to changing conditions. Therefore, the development of adaptive application schedulers has become a major challenge [3, 4,]. Research projects, such as AppLeS [5] and Nimrod/G [6], have demonstrated that periodic evaluation of the schedule in order to adapt it to changing grid conditions and application dynamic demands can result in significant improvements in performance. The Application Level Scheduling (AppLeS) project primarily focuses on developing scheduling agents for individual applications. Thus, the AppLeS framework contains templates that can be applied to problems that are structurally similar and have the same computational mode. Templates have been developed for parameter sweep [7] and master/slave [8] type applications. Nimrod/G is a grid resource broker that provides support for formulation of parameter studies on computational grids as well as facilities for resource

[*] This material is based upon work supported by the National Science Foundation under Grant No. 0424546.
Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

discovery and scheduling. Prophet [9] is an automated scheduler for data parallel applications that utilize a performance model for predicting application performance on different resource combinations. Gallop [10] is a wide-area scheduling system that implements scheduling models across different sites.

3. An Adaptive Wide Area Large Scale Scheduling Framework

The proposed scheduling strategy for grid environments takes into account the following issues. First, scheduling must be guided by Quality of Services (QoS) criteria to get a better match between applications and resources. Second, resources provide non dedicated services to the applications so mechanisms to predict computation time and probability of failure are required. Third, the scheduling algorithm must be flexible enough to allow adaptive reconfiguration of the scheduler components.

3.1 Wide Area Scheduling Model

We assume that the resources are connected via two-level hierarchical networks. The first level is a wide area network that connects local area networks at the second level. Users submit job specifications with their QoS requirements. The scheduler then discovers appropriate resources for processing the job and schedules the tasks on the resources. In order to discover suitable resources, the scheduler has to predict execution times on the available resources and verify QoS capabilities and availability of the resources. Re-scheduling mechanisms are then implemented to adapt scheduling to service dynamics.

3.2. A New Urgency Criterion

Our scheduling strategy focuses on providing high priority to jobs with low probability of failure. To achieve this, an urgency criterion is introduced to account for relevance, laxities and probability of failures of incoming jobs. The proposed urgency criterion is a combination of one static parameter and two dynamic parameters. These parameters are defined as follows.

1. Criticity (Relevance). This static factor is initially established by the user according to experience and/or customer importance. Criticity values range between 0 and 100.

2. QoS (Quality of Service). Scheduling involves matching of job needs with resource availability and capability and addressing the concern of the quality of the match. Different QoS metrics can be defined. For instance, the desirable bandwidth for the application or the required speed of processors

3. Laxity. This dynamic factor is defined as *Laxity* = *Jd* - (*t* + *Jlat*)*;* where *Jd* is the Job deadline, *t* is the actual time of calculation, and *Jl* is the expected latency of the Job. Laxity as defined above is not bounded and may conduct to unrealistic urgency criteria values. One way to compensate this is to define a modulator factor *K* (units of time) as defined in [11].

3.3. The QB-MUF Algorithm

The QB-MUF (Quality of Service Based Maximum Urgency Factor) algorithm iteratively assigns jobs to resources by considering resource availability and job urgency factors. If there are not resources available to process a job, the job is sent to a queue. The urgency criteria of the jobs in the queue are updated with certain periodicity to assure the flow of jobs with high probability of success. The general algorithm can be viewed as follows

```
While (There are jobs to schedule)
        if (There are available resources)
                for each job i to schedule
                        calculate job urgency;
                        allocate job;
                end for
        else
                if (Queue.length >0 )
                        update urgency factor and
                verify QoS
                end if
                insert ordered job to the queue
                re-schedule jobs
        end if
end while
```

3.4. Framework Architecture

The proposed urgency criterion and scheduling algorithm are embedded into a dynamic scheduling framework. The main goal of this framework is to provide a reusable infrastructure to design and evaluate scheduling strategies. An important feature is that the framework has the flexibility to allow adaptive components. The Resource Manager gets the resource load information from a profiler placed inside of each local network. The profiler also maintains a forecasted load and calculates a tuning load. Such a tuning load is calculated from the real load and the forecasted load. The Global Scheduler uses the information gathered by the Resource Manager and static job information to generate scheduling events. A Job Monitor interacts with the Exchange Dispatcher and the Resource Manager to verify that the estimated conditions for scheduling are appropriate. If everything goes fine, the Dispatcher sends the job to its next step through the Workflow Engine. On the other hand, in the case of a failure, a contingency or a change in the schedule conditions occurs, the Dispatcher has to redirect the job according to predefined policies to manage exceptions. To reduce system overload, Local Schedulers are implemented at each local network to deal with local allocation of jobs over available resources.

4. Experimental Results

We use GridSim [12] as simulation tool to implement and evaluate the QB-MUF algorithm. GridSim is a Java-based discrete-event grid simulation package, which allows modeling and simulation of entities in parallel and distributed computing systems. We took advantage of the GridSim capability for implementing new scheduling policies to deploy our QB-MUF algorithm and evaluate its performance.

Throughout the experimentation we compare the behavior of our QB-MUF algorithm with respect to two other scheduling approaches: The Minimum Laxity First, denoted as Laxity, and the well known First In First Out (FIFO) scheduling algorithm. Two metrics were observed throughout the experimentation. First, the number of successful jobs delivered. Second, the mean waiting time of successful delivered jobs. The whole set of experiments ran under the same conditions, except in those explicitly mentioned cases. Job arrival is a draw from an exponential distribution while, while the job sizes follow a normal distribution. Failures are induced to jobs so that QoS requirements may be eventually violated. The first probability of failure ranges from 40% to 55% and the second failure ranges between 20% and 40%. The joint probability ranges between 50% and 70%.

Figure 1 shows the order of execution of jobs for the three scheduling algorithms. Notice that for QB-MUF jobs with high QoS deliveries are first executed. Figure 2 shows the results for a total of 1000 jobs with arrival rate of 0.35. The mean processing time is defined as the average of the time that a job has to wait since it was received until its start processing. Results show a reduction of waiting processing time of the QB-MUF over laxity and FIFO approaches. QB-MUF exhibits a better behavior in terms of the number of successful jobs over time compared to the traditional approaches. We point out that for the experiments illustrated here the advantage of the QB-MUF algorithm is more evident around specific units of time (for example 16,000 units of time in the case illustrate in Figure 2). This is because QB-MUF gives more importance to those jobs that have god expectations of finishing successfully.

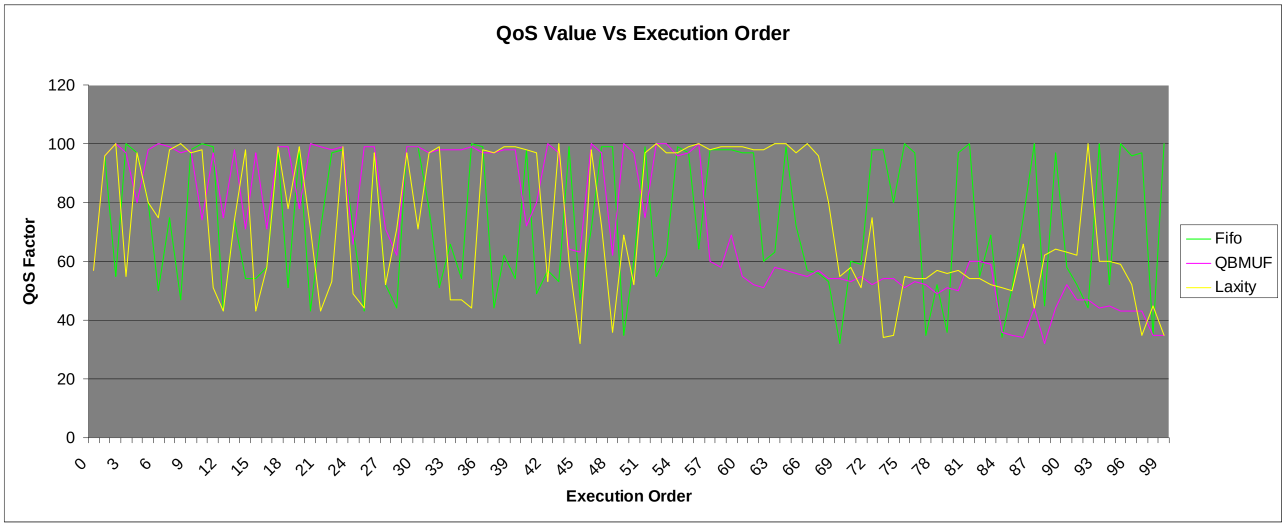
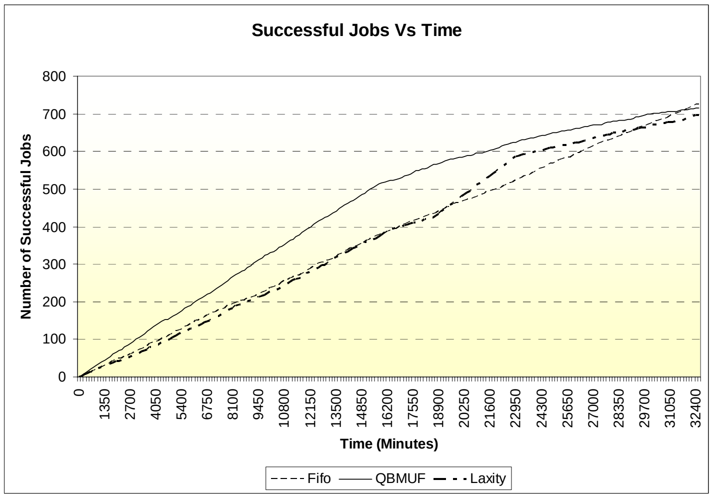


Figure 1: Quality of Service Guided Execution Order; jobs=100; arrival rate=0.35



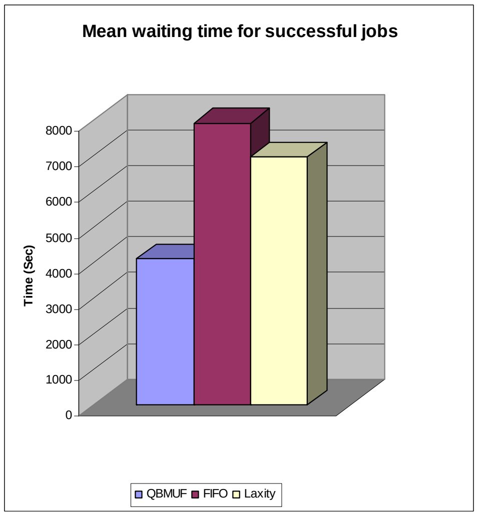


Figure 2: (a) Number of successful jobs; (b) Mean waiting time; jobs=1000; arrival rate=0.35

5. References

[1] I. Foster and C. Kesselman, Eds. The Grid: Blueprint for a New Computing Infrastructure. Morgan-Kaufmann, 1999.

[2] F. Berman, G. Fox, and T. Hey, Eds. Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons, 2003.

[3] F. Berman and R. Wolski, Scheduling from perspective of the application. *Proceedings of the Symposium on High Performance Distributed Computing*, 1996.

[4] J.M. Schopf, Ten actions when superscheduling. *Technical Report WD8.5*, Global Grid Forum, 2001. Scheduling Working Group.

[5] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov, Adaptive Computing on the Grid Using AppLeS *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 14(4), 369-382, 2003.

[6] R. Buyya, D. Abramson, and J. Giddy, A computational economy for Grid and its implementation in the Nimrod/G resource broker. *Future Generation Computer Systems*, Elsevier Science, 2002.

[7] H. Casanova, G. Obertelli, F. Berman, and R. Wolski, The AppLes parameter sweep template: User-level middleware for the Grid. *Proceedings of Supercomputing 00*, 2000.

[8] G. Shao, R. Wolski, and F. Berman, Master/slave computing on the Grid. *In Proc. Heterogeneous computing Workshop*, 2000.

[9] J.B. Weissman, “Prophet: Automated scheduling of SPMD programs in workstation networks.” Comcurrency: *Practice and Experience*, 11(6), 1999.

[10] J.B. Weissman, Gallop: The benefits of wide-area computing for parallel processing. *Journal of Parallel and Distributed Computing*, 54(2):183-205, 1998.

[11] W. Lozano and W. Rivera, A scheduling framework applied to digital publishing workflows. Proceedings SPIE Vol. 6076, 198-209, Digital Publishing, 2005.

[12] A. Sulistio, G. Poduvaly, R. Buyya, and C. Tham, Constructing a grid simulation with differentiated network service using gridsim, *Proceedings of The 6th International Conference on Internet Computing (ICOMP'05)* , 2005.