QoS Oriented Resource Reservation in Shared Environments

Ming Wu, Xian-He Sun, Yong Chen
Department of Computer Science
Illinois Institute of Technology
Chicago, Illinois 60616, USA
(wuming, sun, chenyon1@iit.edu)

Abstract

Resource sharing across different computers and organizations makes it possible to support diverse, dynamic changing resource requirements of distributed applications. Reservation mechanisms have been used to reserve resources for external applications through service level agreements between local resource organizations and external applications. However, the effects of resource reservation on local applications, and therefore the trustfulness of the successful fulfillment of the service agreement, have been ignored. In this paper, we investigate the effect of resource reservation on external applications as well as local jobs, and design efficient task scheduling algorithms considering the tolerance of local jobs to resource reservation. Extensive simulations and implementation experiments have been carried out to confirm our analysis results. Experimental results show that the relative slowdown metric and the failure-minimization scheduling algorithms proposed in this study are practically effective and have a real potential.

1. Introduction

Resource sharing has become common place in distributed computing. The sharing exists not only in computation resources but also in communication resources and storage resources. In Grid computing, virtual organizations coordination has been proposed to enable resource sharing among disparate groups of organizations and individuals [1].

A challenging issue, however, arises in this new situation: how external applications (i.e. applications that are being managed by external organizations or individuals) and local applications (i.e. jobs that are managed by the local resource administration domain) share resources appropriately. To satisfy external applications' required Quality-of-Service, reservation is proposed as a mechanism of resource sharing among external applications and local jobs.

Some fundamental problems still exist: how much and how long resources should be reserved for external applications? If little resource is reserved, the external application may have to run for a long time. If resource is over reserved, the performance of local jobs may be greatly affected. Recent research [2][3] has shown that local users often power cycle the resource immediately if uncomfortable machine slowness is observed. This indicates that an inappropriate resource reservation will result in a failed fulfillment of the service level agreement. The lack of a unified analysis of the performance impact of resource reservation on local jobs and external applications prevents the adoption of resource reservation among individuals and organizations and thus the appropriate delivery of resource sharing in distributed computing, especially in Grid computing.

In this study, we propose a new performance metric, the relative slowdown, to quantify the performance impact of resource reservation. We model the local job process with an M/G/1 queuing system and analyze the effect of system parameters on relative slowdown. We investigate both first-come-first-serve (FCFS) and round-robin (RR) queuing disciplines. Efficient algorithms are designed and implemented considering local jobs' tolerance to reservation. A user-level soft real time CPU scheduler, DSRT, is updated to enable resource reservation in a general computing platform. Extensive simulations have been carried out to verify our analytical results, and experimental studies have been conducted to confirm the correctness and effectiveness of the newly proposed metric in real production systems.

The rest of this paper is organized as follows. Section 2 reviews some related work. Section 3 introduces the proposed relative slowdown metric and presents the analytical results. Task scheduling algorithms under reservation are then given. Section 4 presents implementation details of CPU reservation

and system parameter measurement. Section 5 presents the simulation and experimental results. Finally, in Section 6, we summarize the current work and discuss future work.

2. Related work

Several sharing policies have been deployed in distributed systems. In SETI@home, Entropia, and Condor, an external application is allowed to run only when no keyboard and/or mouse activities are detected on local resources. This sharing mechanism ensures that external applications do not interfere with the execution of local jobs, but it cannot provide the QoS guarantee for external applications. Another sharing mechanism [4] is that external applications compete with local jobs for resource occupancy. This sharing mechanism cannot assure the desired level of QoS of applications either. The third sharing policy is that resources are reserved for the execution of external applications to satisfy their QoS requirements [5]. In Grid computing, resource reservation is enabled with the General-purpose Architecture for Reservation and Allocation (GARA) [6].

Scheduling methodologies in shared environments have been widely studied. AppLeS [4] is a well-known task scheduling system for Grid computing. Scheduling algorithms in the AppLeS are supported by short-term resource availability prediction provided by NWS services [7]. In contrast, a long-term, application-level performance prediction and task scheduling system, namely Grid Harvest Service (GHS) system, is proposed recently [8]. Both AppLeS and GHS scheduling provide the estimate of application completion time. However, neither of them can guarantee that an application will be finished for a given deadline because resource availabilities may vary during the time. Virtual Application Service (VAS), a deadline-bound system, makes scheduling decision based on resource reservation [10]. According to the application information, VAS determines the CPU resources to support the application QoS. However, the scheduling decision is made only based on the application QoS requirement. It doesn't consider the impact of reservation on local jobs.

Research in performance impact of resource reservation has attracted increasing attention recently [11][12]. J. Cao et al implement a light weight job scheduler, COSY, to support both queue scheduling and advance reservation [11]. They carry out experiments to observe the impact of advance reservation on the mean waiting time, resource utilization, and the reject rate. The performance impact of resource provisioning on the completion time of workflows has been studied by G. Singh et al [12]. Their simulation results have shown that the workflow completion time is reduced by 50% using reservations. Their work has demonstrated the importance of the analysis of performance impact of reservation in scheduling algorithm designing. However, they are observational in nature. There is no analytical result on the reservation impact on local jobs and external applications.

3. Performance modeling and analysis

We call an external application the remote task. In order to identify the impact of resource reservation on local jobs and the remote task respectively, we build up a model to describe the local job process. The arrival of the resource owner's local job is assumed to follow a Poisson distribution with rate λ. The execution time of local jobs is assumed to follow a general distribution with mean $1/\mu$ and standard deviation c . μ is also called the service rate. The job arrival assumption is based on the observations of hyperexponential machine usage patterns reported by researchers in Wisconsin-Madison, Berkeley, Maryland and et al [13]. The general service assumption is a generalization of the observed machine usage pattern. Based on our assumption, the resource owner's local job process is an M/G/1 queuing system. Notice this model has been widely used in literature [9][14].

3.1. Effects of resource reservation on local jobs and the remote task

To make an appropriate resource reservation, we need to identify the effect of resource reservation on the performance of local jobs. In queuing systems, an often used performance measure is the mean waiting time, which is defined as the average waiting time of local jobs [15]. Two waiting time metrics are widely used in the literature. One is the waiting time in queue. Another is the waiting time in system. Because it is easier for users to feel and measure how long their jobs are finished than how much time their jobs may stay in a "queue", we choose waiting time in system to investigate the impact of resource reservation on local jobs and simply call it waiting time throughout the study.

When a certain part of CPU resource is reserved for an external application, the CPU resource available to local jobs is reduced. As a result, the local job's waiting time will be increased in general. To determine whether a reservation is accepted or not, local users are naturally concerned about how much the average waiting time is affected. Let W_a denote the mean

waiting time of local jobs after reservation and W_b denote the mean waiting time of local jobs before reservation. We can use either $W_a - W_b$ or W_a / W_b to describe the change of the average waiting time. The latter is a more appropriate performance metric for our study since it gives the relative variation of the waiting time. The former reflects the absolute increased value of the waiting time. It can be used to compare the effects of different reservation quotas on local job performance at one resource. It may be insufficient, however, to compare the effects of the same reservation quota for different resources with different local job demands. For example, 10-second increased waiting time caused by reservation indicates a lot of performance change for local jobs at the average of 10-second waiting time but a little for local jobs at the average of 200-second waiting time. This is because users are more likely to anticipate short delays for small jobs and are willing to tolerate longer delay for large jobs [19]. Based on the above discussion, we introduce a new performance metric, relative slowdown (S_R) to reflect the impact of resource reservation on local jobs.

Definition 1 The *relative slowdown* of local jobs on a resource for a given reservation is the ratio of the average waiting time with reservation and the average waiting time without reservation.

We name the new performance metric as the relative slowdown because W_a / W_b has a similar format as the performance metric in the queuing system, slowdown, which is defined as the ratio of the waiting time and the job's workload. In this study, we focus on the reservation of CPU resource. We use κ to represent the reserved part of CPU resource. We call it the reservation ratio.

Two queuing disciplines are widely used in a general computer system for choosing which job in the queue is to be serviced next. They are first-come-first-serve (FCFS) and round-robin (RR). In a FCFS queue, jobs are served in the order of their arrival times. After one is finished, the next one will be served. In a RR queue, each job in the queue is served in a quantum of time and all jobs are served in turn.

Theorem 1. Given a M/G/1 FCFS queuing system where the local jobs' arrival rate is λ . Let κ be the reservation ratio. Then the mean waiting time after reservation and the relative slowdown, are

$$W_a = \frac{1}{1 - \kappa} \varphi(\lambda, \rho, \sigma, 1-\kappa) \lambda^{-1} \qquad (1)$$

$$S_R = \frac{1}{(1-\kappa)} \left(\frac{\varphi(\lambda, \rho, c, 1-\kappa)}{\varphi(\lambda, \rho, \sigma, 1)} \right) \qquad (2)$$

where $\varphi(\lambda, \rho, \sigma, c) = \rho + \frac{\rho^2 + \lambda^2 \sigma^2}{2(c-\rho)}$. $\rho = \lambda / \mu$ is machine utilization and c is the service time standard deviation.

Please note that $1 - \rho > \kappa > 0$ holds in the above equation. Otherwise the system cannot arrive at a steady state after reservation and the average waiting time will be infinite.

Theorem 2. Given a M/G/1 RR queuing system where the local jobs' arrival rate is λ . Let κ be the reservation ratio. Then the mean waiting time after reservation and the relative slowdown, are

$$W_a = \frac{\rho}{\lambda(1-\kappa-\rho)} \qquad (3)$$

$$S_R = \frac{1 - \rho}{(1-\kappa-\rho)} \qquad (4)$$

The proof of Theorem 1 and 2 is given in [16].

We have introduced the S_R metric to describe the impact of resource reservation on local job completion time. What is the effect of reservation on the remote task execution time is another interesting problem. Let ϖ denote the workload of the remote task, T denote the completion time of the remote task and τ denote the resource computing capacity of a resource where the remote task executes. Since the reserved CPU resource is dedicated to the execution of the remote task, the task completion time can be expressed as

$$T = \frac{\varpi}{\kappa \tau} \qquad (5)$$

On the other hand, if a remote task has a requirement of the completion time, T , we can calculate the correspondent reservation ratio, κ , using

$$\kappa = \frac{\varpi}{T \tau} \qquad (6)$$

3.2 Effect of system parameters on relative slowdown

Using formula (2), we examine the effects of different system parameters on the relative slowdown in an M/G/1 FCFS queuing system. Four parameters are examined in our computations: λ , c , ρ , and κ . Their default values are set as: $\lambda = 1$, $c = 0.8$, $\rho = 0.2$, $\kappa = 0.2$.

Figure 1 (a) gives the relative slowdown for different utilizations and different reservation ratios. We observe that S_R increases when the reservation ratio, κ , increases given a fixed utilization. When κ is close to $1 - \rho_b$, S_R is increasing dramatically and

tends to be $\propto$ (we limit the maximum value of S_R as 30 in the figure). Figure 1 (a) also shows that, for a given reservation ratio, S_R increases when the utilization, ρ_b, increases. When ρ_b is close to 1- κ, S_R is increasing dramatically and tends to be $\propto$. Figure 1(b) shows the effect of κ and λ_b on S_R. For a given λ_b, S_R increases when κ increases and tends to be $\propto$. For a given κ, S_R also increases when λ_b increases. However, it increases too slowly to differentiate it from the graph. This indicates that the change of λ_b does not have the same impact on S_R as κ does. We find the variation of c_b has a similar impact on relative slowdown as that of λ_b, so we don't give its graph here. Figure 1 (c) gives the variation of S_R for different combinations of λ_b and c_b. It shows that there is a maximum S_R (in this case it is 1.67) whatever λ_b and c_b are when κ and ρ_b are fixed. Based on these graphs, we conclude that the utilization and the reservation ratio are two dominant parameters in determining S_R for an M/G/1 FCFS queuing system. Figure 1(d) gives the relative slowdown for different utilizations and different reservation ratios in an M/G/1 RR queuing system. It presents the same pattern as Figure 1(a). However, the curves in Figure 1(d) are relatively steeper than those in Figure 1(a). For the same utilization and reservation ratio, the relative slowdown tends to be smaller. This indicates a RR queuing discipline more favors reservation than a FCFS queuing discipline.

3.3. Task scheduling under resource reservation

In scientific computing, applications often require real-time, deadline-bound execution. Resources are reserved in advance for application running so that the application can be completed before its deadline. In current reservation-based scheduling strategies [10][12], CPU resources are allocated and reserved only based on the application's deadline constraints. The effect of resource reservation on local jobs' performance is not considered. Consequently, the potential task scheduling failure is ignored.

We propose failure-minimization scheduling to address this problem. The goal of failure-minimization scheduling is to minimize the reservation failure rate while satisfying the deadline requirement of the remote task. In this study, we assume that the reservation failure rate is in proportional to the users' discomfort probability, which is the probability that a user will

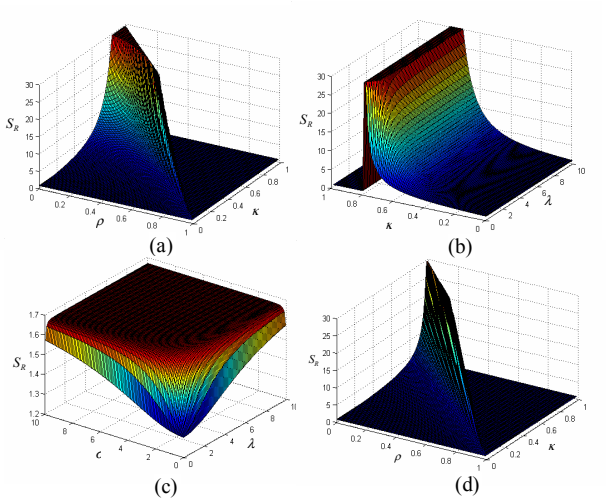

Figure. 1. Effect of parameters on relative slowdown

feel discomforted when reservation occurs [2]. So the goal of failure-minimization scheduling is turned to minimize the users' discomfort while satisfying the deadline requirement of the remote task.

Let us suppose that we have a list of machine, $M = \{m_1, m_2, \ldots m_q\}$ and each machine m_k has a cumulative probability distribution (CDF) of discomfort in terms of relative slowdown, $B_k(x) = P_k[S_R \le x]$. The CDF can be obtained based on the measurement history. The remote task with a workload of ϖ has a deadline of T_D for its execution. The single sequential task scheduling problem for failure-minimization can be formulated as finding a machine m_i where $B_i(S_{Ri})$ is the minimal and T_i is less than or equal to T_D. T_i is the remote task completion time and S_{Ri} is the correspondent relative slowdown on machine m_i. According to formula (2) and (4), to get a minimal relative slowdown, the reservation ratio should be as small as possible. However, κ is bounded by the deadline constraint. So the minimum κ_i is decided by

$$\kappa_i = \frac{\alpha}{T_D \tau_i}. \qquad (7)$$

After identifying the reservation ratio, we can calculate relative slowdown with formula (2) and (4). Then we can use the given CDF of discomfort to find the correspondent probability of discomfort on each resource. We will allocate the remote task to the machine which has the minimum probability of discomfort and make advance reservation on it.

The successful running of a parallel application requires the successful fulfillment of resource

reservation on each machine where the subtask of the application is running. Suppose that the probability of reservation failure on each resource is independent. The probability of successful reservation for the parallel application is the production of the probability of successful reservation on each resource. Since we use user's discomfort to represent the reservation failure rate, we define a metric of global discomfort as $G_D = 1 - \prod_{\cdot} (1 - B_i(S_{Ri})$ to reflect the failure probability of resource reservation for the whole application. Our goal of parallel task scheduling is thus to minimize the global discomfort. We assume that the remote task can be arbitrarily partitioned and the CDF of discomfort on each resource is independent. The parallel task scheduling problem for failure-minimization can be formulated as finding a machine set $M_s = \{m_{l1}, m_{l2}, \ldots m_{\ln}\}$ where the global discomfort is the minimal and T_{lk} is less than or equal to T_D. If we assume that the unit of execution time and application workload is one-second, we can use dynamic programming to solve this problem. However, it is too costly when the application workload or the system scale is large. A heuristic task-scheduling algorithm in Figure 2 is thus proposed. From the definition of global discomfort, we can see a machine set with fewer machines may lead to a smaller G_D. However, the increased workload on each machine would require a larger reservation ratio to satisfy the deadline constraint and thus lead to a bigger G_D. This tradeoff indicates machines should be selected carefully and priority should be given to those machines which can take more application workload with less user's discomfort. We define a metric of safe workload as $\alpha_s = T_D * \kappa_{0.1} * \tau$ to reflect how much workload of an external application can be supported by a resource with a user's discomfort probability of 0.1. $\kappa_{0.1}$ is the correspondent reservation ratio for a relative slowdown which leads to a user's discomfort probability of 0.1. The heuristic algorithm includes two basic steps. The first step of this algorithm is to sort each lightly loaded machine with α_s. A higher value α_s of a machine indicates this machine has more available computing power for an application and thus should be considered first. The second step of this algorithm is to use the bi-section search to find the local optimal based on the ordering. To partition the application workload among a given machine set $\{m_{s1}, m_{s2}, \ldots m_{sj}\}$, we apply an equal-discomfort partition strategy. That is, each machine is assigned with a workload so that the probabilities of discomfort on different machines are the same. It is based on the intuition that the production of probabilities of user's comfort $(1 - B_i(S_{Ri}))$ tends to be maximum if each of them is the same. Let β denote this discomfort probability. Then $B_{sk}^{-1}(\beta)$ is the correspondent relative slowdown on machine m_{sk} and the reservation ratio is

$$\kappa_{sk} = S_{Rsk}^{-1}(B_{sk}^{-1}(\beta)). \quad (8)$$

Since $\varpi = \sum_{k=1}^{j} \omega_{sk}$ and $\omega_{sk} = T_D * \kappa_{sk} * \tau_{sk}$, we have

$$\varpi = \sum_{k=1}^{j} T_D * S_{Rsk}^{-1}(B_{sk}^{-1}(\beta)) * \tau_{sk}. \quad (9)$$

β is the solution of formula (9). After calculating β, we can use formula (8) to get the reservation ratio on each machine.

Assumption: a remote task can be partitioned into any size of sub-tasks. Each sub-task will be assigned to a machine respectively.

Begin

List a set of idle machines that are lightly loaded over an observed time period, $M = \{m_1, m_{2,} \ldots m_q\}$;

Sort the list of idle machines in a decreasing order with ω_s, $M' = \{c_1, c_{2,} \ldots c_q\}$;

$a = 1$, $b = q$;

Repeat

$c = \lfloor (a+b)/2 \rfloor$

/* $f(x)$ denotes $G_D(C(x))$ where $C(x) = \{c_1, c_{2,} \ldots c_x\}$ */

If $f(a) = \min\{f(a), f(b), f(c)\}$ **then** $b = c$

Else If $f(b) = \min\{f(a), f(b), f(c)\}$ **then** $a = c$

Else If $f(c) < f(c+1)$ **then** $b = c$

Else $a = c$

Until $a + 1 = b$

If $f(a) < f(b)$ **then**

Assign parallel task to the machine set $C(a)$;

Else Assign parallel task to the machine set $C(b)$;

End

Figure 2. Heuristic task scheduling algorithm

4. Implementations

In this study, we apply the DSRT 2.0 software for CPU reservation. DSRT is a dynamic soft real time scheduler and has been widely used in Grid communities [17]. DSRT is a part of the QualMan (QoS-aware resource management) middleware, which

consists of a set of resource servers (schedulers and brokers) to provide QoS, negotiation, admission, and reservation capabilities for sharing resources such as CPU, network, and memory. It was designed to support QoS requirements of distributed multimedia applications.

DSRT is the implementation of the CPU server of QualMan middleware. A major component of DSRT is the resource scheduler, named the dispatcher. A priority scheduling mechanism is applied to differentiate the processing of real-time (RT) processes and time-sharing (TS) processes. The dispatcher runs at the highest fixed-priority. It wakes up periodically to dispatch RT processes by moving them between the waiting priority (the lowest fixed-priority) and the running priority (the second highest fixed-priority). When the dispatcher sleeps, the RT process with the running priority is scheduled. When no RT processes exist, the dispatcher yields its CPU control and then TS processes are executed using the fair time-sharing scheduler of UNIX. In the current DSRT Linux implementation, the real-time process has to call the yield API to explicitly generate the yield signal. In this work, we treat the external application as a RT process. We monitor the resource usage of the external application. When the required resource reservation is satisfied during each time slot, a yield sign is raised and the external application is sent to a waiting queue until the start of the next time slot. Figure 3 gives the basic flowchart of the dispatcher.

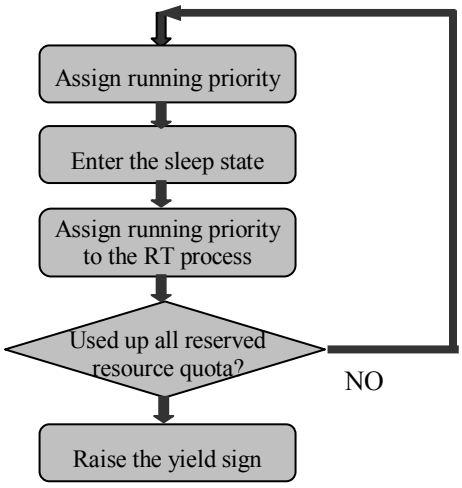

Figure 3. A flowchart of dispatcher

5. Experimental Results and Analysis

Experimental testing has been conducted to verify the correctness of the proposed relative slowdown metric and the associated analytical results with extensive and rigid simulations. We also carry out experiments on a real system with an updated DSRT, based on trace files collected from the real environments. Experimental results show that the proposed relative slowdown is practically applicable.

We build a simulation model of an M/G/1 queuing system. The model is composed of a local job generator, a waiting queue, a scheduler, and a server. The local job generator generates local job traffic that follows Poisson arrival and different service time distributions. In our simulation model, we support three types of service time distributions: Exponential distribution, Gamma distribution, and Pareco distribution. When a job arrives, it first enters the waiting queue and stays until the server is available. We assume that there is no limitation on the job queue size. The scheduler decides how to schedule jobs in the waiting queue. Two queuing disciplines are supported in our model: FCFS and RR. Each job in the RR queue is scheduled to occupy the CPU server for a fixed time slice (the default value is 0.0001 second) in turn.

In our simulation, we first examine the impact of reservation on local jobs' performance. The simulation runs 50 times for each parameter set. Figure 4 gives the variation of relative slowdown with different utilization and reservation ratios for Exponential distribution. Gamma, and Pareco distribution present the same pattern. Compared with the analytical results given in Figure 1 (a), we find that the simulation results present a similar performance impact of resource reservation. To further evaluate the correctness of the analytical results given in formula (2) and (4), we test the prediction error in the simulation. Figure 5 plots the mean and variation of percentage prediction error with different reservation periods from 2000 seconds to 10000 seconds when the reservation ratio is 0.2 and the utilization is 0.15. The percentage prediction error is defined as $\left|\frac{\mathrm{Pr}\,ediction - Measurement}{Measurement}\right|$ where the predicted value is calculated by the formula (2) and (4) and the measured value is collected from simulation results. The top graph in Figure 5 shows the mean of prediction error and the bottom one gives the standard deviation of prediction error. We observe that

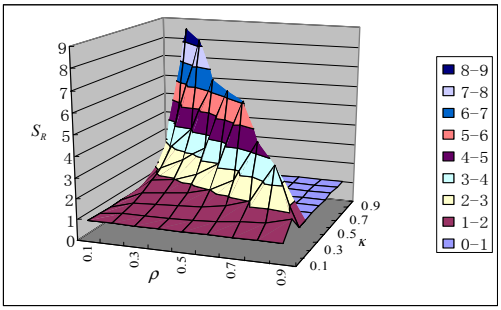

Figure 4. The variation of relative slowdown with different utilizations and reservation ratios

both mean and variation are very small with the three distributions. The longer the reservation period, the

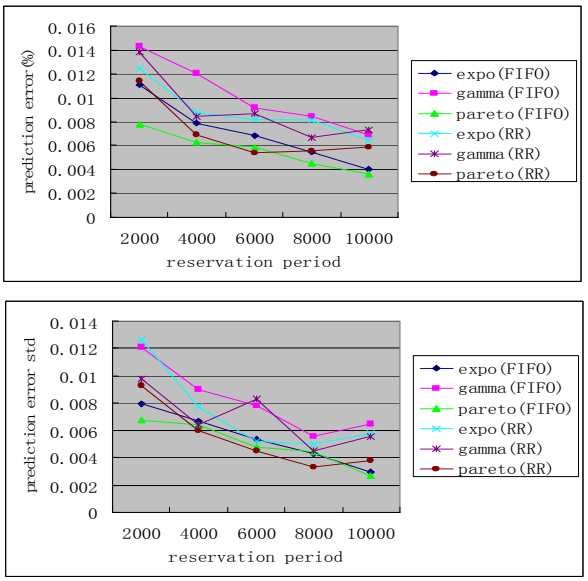

Figure 5. The mean and variation of prediction error with different reservation periods

smaller the mean and variation of the prediction error. In our simulation, we also examine square prediction error, which is defined as $(\Pr ediction - Measurement)^2$. It presents similar results. These results demonstrate the correctness of the analysis of relative slowdown.

Simulations are also conducted to test the efficiency of the proposed heuristic failure-minimization scheduling algorithms for parallel processing. The machine set sizes are 20, 30, and 40 respectively. We use Weibull distribution to describe the user's discomfort probability since it is often used to describe failure rate in practice. Notice our scheduling can be applied with any empirical distributions. The shape and scale parameters vary on each machine. We compare the performance of the five scheduling strategies: heuristic, random, fast-speed, light-load. Fast-speed and light-load scheduling means that machines are selected based on their speeds and loads respectively. For example, Fast5 represents the first 5 fastest machines are selected for task allocation. Figure 6 gives the 5th, 50th, and 95th percentiles of failure rate (the global discomfort) over 100 simulations for each scheduling policy. We can observe that not only the average reservation failure rate with heuristic scheduling is much lower than those with other scheduling algorithms, but also the variation of the failure rate with heuristic scheduling is rather smaller than others.

To test the applicability of the proposed relative slowdown and the efficiency of the scheduling algorithms in practice, we conduct experiments on a real system with the updated resource reservation tool, DSRT. The performance measure mechanisms discussed in Section 4 are used to measure system parameters. The experimental platform is the IIT (Illinois Institute of Technology) cluster of the DOT Grid Testbed [18]. It contains one server and 13 computing nodes. The server has two 2.4GHz CPUs and 2.5 GB main memory, and each node has four 2.4GHz CPUs and 2.5 GB main memory. The operating system is Linux 2.4.20-8.

The lifetime of local jobs is simulated with 2.0/x where x is a randomly generated number between 0 and 1. This distribution follows the observation of real-life processes [19]. Two classes of external applications are used in the experiments. One is meta-task, which consists of a set of independent indivisible subtasks. A typical example of meta-task is the parameter sweep application, a widely used grid application [4]. In the test, we generate a synthetic meta-task. Each of its subtasks is a computation intensive program. Another is parallel program. We use Cactus parallel application, a numerical simulation of a 3D scalar field [20]. In the experiment, the server and three nodes, iit01, iit02, and iit03 are used. The utilization on these machines is set as 0.2 and the reservation ratio as 0.2. FCFS queuing discipline is enforced on each resource. Table 1 gives the actual CPU part occupied by the external applications and the measured relative slowdown of local jobs. It shows that the measurement is close to the analytical result (the calculated relative slowdown is 1.33). We notice that Cactus occupies less resource than expected. This might be due to the synchronization and communication among processes.

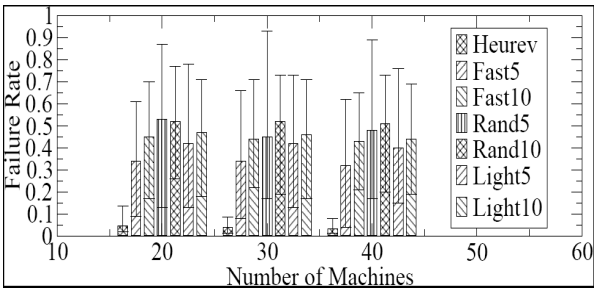

Figure 6. Failure rates with different scheduling algorithms

Table 1. Performance of external applications and local jobs under reservation

Measurements		Reservation periods (hours)			
		1	2	4	8
Meta-task	CPU occu.	20%	20%	20%	20%
	S_R	1.35	1.29	1.30	1.33
Cactus	CPU occu.	19.1%	18.3%	17.6%	17.9%
	S_R	1.27	1.37	1.38	1.35

5. Conclusion

Resource reservation is an effective mechanism to warrant QoS of the remote tasks in a shared environment. While much effort has been made on how to build the service agreement between local resource organizations and external applications, the influence of resource reservation on local jobs is not well understood. The research of appropriate scheduling and management of resource reservation is still in its infancy.

In this study, we first introduce a relative slowdown metric to measure the slowdown of resource reservation on local jobs. Using results in queuing system, we next derive formulae for calculating the proposed relative slowdown metric. Based on the derived formula, we examine the effects of different system parameters on the relative slowdown. We also investigate the effect of reservation on the completion time of external applications. After identifying the impact of resource reservation on both local jobs and external applications, we present sequential task scheduling and parallel task scheduling algorithms under resource reservation. Finally, we conduct extensive and rigid simulations to verify the correctness of the relative slowdown metric and the scheduling algorithms. The experimental results match the analytical results well. We also conduct experiments on production systems with an updated resource management tool, DSRT, under the DOT Grid environment. Experimental results show our study provides a suitable solution to balance the need of remote and local users in a shared environment. We plan to explore the potential of these scheduling strategies further in our future work to fully embed them into Grid computing environments.

Acknowledgment

This research is supported in part by national science foundation under NSF grant ACI-0305355, EIA-0224377, ANI-0123930, and EIA-0130673.

Reference:

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, 2nd Edition, Morgan-Kaufman, 2004.

[2] Gupta, B. Lin, P. Dinda, "Measuring And Understanding User Comfort With Resource Borrowing," *in: Proc. of the 13th IEEE International Symposium on High Performance Distributed Computing*, Honolulu, Hawaii, 2004.

[3] D. Nurmi, J. Brevik, and R. Wolski, "Modeling Machine Availability in Enterprise and Wide-area Distributed Computing Environments," *UCSB Computer Science Technical Report (CS2003-28)*, 2003.

[4] F. Berman, R. Wolski, H. Casanova, W. Cirne, et al. "Adaptive computing on the Grid using AppLeS," *IEEE Trans. Parallel Distrib. Systems*, 14 (2003) pp. 369-382.

[5] R. J. Al-Ali, K. Amin, G. V. Laszewski, O. F. Rana, D. W. Walker, M. Hategan, N. Zaluzec, "Analysis and Provision of QoS for Distributed Grid Applications," *J. Grid Comput*. 2(2) (2004) pp. 163-182.

[6] I. Foster, A. Roy, V. Sander, "A quality of service architecture that combines resource reservation and application adaptation**,"** *in: Proc. of The* International Workshop on Quality of Service, June 2000, pp. 181-188.

[7] R. Wolski, N. T. Spring, J. Hayes, "The network weather service: a distributed resource performance forecasting service for metacomputing," *J. Future Generation Computing Systems*, 15 (1999) pp. 757-768.

[8] X.-H. Sun, M. Wu, "Grid Harvest Service: A System for Long-Term, Application-Level Task Scheduling," *in: Proc. of 2003 IEEE International Parallel and Distributed Processing Symposium,* Nice, France, April 2003.

[9] L. Gong, X.-H. Sun, Edward F. Waston, "Performance modeling and prediction of non-dedicated network computing," IEEE Trans. Comput. 51 (2002) pp. 1041-1055.

[10] K. Keahey and K. Motawi, "The Taming of the Grid: Virtual Application Services," *Technical Memorandum ANL/MCS-TM-262,* May 2003.

[11] J. Cao and F. Zimmermann, "Queue scheduling and advance reservations with COSY," *in Proc. of the 18th International Parallel and Distributed Processing Symposium,* New Mexico, April 2004.

[12] G. Singh, C. Kesselman, E. Deelman, "Performance Impact of Resource Provisioning on Workflows," *in CS Tech report 05-850,* 2005, University of Southern California.

[13] A. Acharya, G. Edjlali, J. Saltz, "The utility of exploiting idle workstations for parallel computation," *in: Proc. SIGMETRICS*, 1997, pp. 225-236.

[14] J. Wei, X. Zhou, and C.-Z. Xu, "Robust Processing Rate Allocation for Proportional Slowdown Differentiation on Internet Servers," *IEEE Transactions on Computers*, Vol. 54, No. 8, (2005) pp. 964-977.

[15] D. Gross, C. M. Harris, *Fundamentals of Queuing Theory*, 3rd Edition, John Wiley & Sons, 1998.

[16] M. Wu, X.-H. Sun, Y. Chen, "Resource reservation in an M/G/1 queuing system," *IIT Computer Science Technical Report (CS2005-02),* 2005.

[17] DSRT2.0, http://cairo.cs.uiuc.edu/software/DSRT-2/dsrt-2.html.

[18] DOT, http://www.dotresearch.org/.

[19] M. Harchol-Balter and A. Downey, "Exploiting process lifetime distributions for dynamic load balancing," *in: Proc. of the 1996 ACM Sigmetrics Conf. Measurement and Modeling of Computer Systems,* 1996, pp. 13-24.

[20] G. Allen, W. Benger, T. Goodale, H.-C. Hege, G. Lanfermann, A. Merzky, T. Radke, E. Seidel, J. Shalf, "Cactus Tools for Grid Applications," *Cluster Computing,* 4 (2001) pp. 179-188.