Chapter 49
QoS-Constrained Resource Scheduling in Grid Computing

Fujian Qin

Abstract Efficient quality of service QoS management is critical for computational grid to meet heterogeneity and dynamics of resources and users' requirements. Aimed at the QoS requirement for resource, QoS-constrained resource scheduling algorithm is proposed. All the tasks are needed to be associated with four QoS dimensions, namely time, reliability, security, and cost. It is implemented and the advantages of the new algorithm are investigated in a grid simulator called Grid Sim after the simulator has been expanded. The results of the simulation experiments show that this new scheduling algorithm effectively achieves load balancing of resources with comprehensive advantages in time efficiency, and solution accuracy compared to the other two algorithms. The approach can reduce data access latency, decrease bandwidth consumption, and distribute storage site load. It can be applied to resource scheduling in grid computing.

Keywords Grid · Quality of service (QoS) · Resource scheduling algorithm

49.1 Introduction

Grid computing is becoming a popular way of providing high-performance computing for many data intensive, scientific applications [1, 2]. It integrates scattered clusters, servers, storages, and networks in different geographic position to form a virtual super computer. Grid computing is a form of distributed system

F. Qin (✉)
School of Software Engineering, Chongqing University of Arts and Sciences, Chongqing 402160, China
e-mail: qinfujian@163.com

Z. Zhong (ed.), *Proceedings of the International Conference on Information Engineering and Applications (IEA) 2012*, Lecture Notes in Electrical Engineering 216,
DOI: 10.1007/978-1-4471-4856-2_49,

where in computing resources are shared across networks, such as high-performance computing power, expensive experiment equipment, and other rare resources [3]. Because of the characters of dynamic, distributed, and different requirements to resources asked by applications running in the Grid Systems, it makes the resource management and scheduling very complex [4]. With the development of grid computing, dealing with the data distribution requires a resource management and scheduling which is faster and more effective for parallel applications in order to reduce task's running time and increases the throughput of the grid system. Hence, resource management and scheduling are integrated components of grid computing, especially the research of task scheduling algorithm, playing a very important role in all the research of grid computing.

In resource management and scheduling, one allocates the submitted manufacturing tasks or resource service requests to appropriate resources within the shortest time and highest benefit to both resource provider and consumer. The resource consumers have various requirements, objectives, and demand patterns, which may require different types and levels of Quality of Service (QoS). Computational grid's resource management and scheduling must deal with various demands from consumers and providers. The way to achieve it is to search the qualified resource service, according to a user's QoS requirements, and realize the mapping from QoS requirements to resource capabilities. Thus, it is important for the grid computing to support scheduling strategy in accordance with QoS policy. The scheduler in the grid environment needs to consider the QoS to get a better match between applications and resources. In this case, QoS-constrained resource scheduling algorithm is proposed and discussed. It turned out that the algorithm implement is a feasible option which takes time and cost constraint and other factors into account. It can allocate application tasks to appropriate resources based on the user's QoS requirement of time, security, reliability, and cost constraint.

49.2 Related Works

Recently, QoS-based resource management and scheduling has been the subject of many research studies. In [1], QoS for network is integrated to make a better match among different levels of QoS request/supply. Dŏgan et al. consider the problem of scheduling a set of independent tasks with multiple QoS requirements [2]. Ding et al. provide a flexible way of QoS-based scheduling in which makespan and service ratio can be traded off by adjusting the preference factor in a local objective function [3]. Lee et al. use resource–utility functions in a QoS management framework with the goal to maximize the total utility of the system [4]. He et al. introduce the matching of the QoS request and service between the tasks and hosts based on the conventional Min–Min algorithm [1]. Li Chunlin et al. defines another three QoS parameters, cost, deadline, and reliability, with utility

function for each dimension and uses a market model to maximize the global utility [5]. Nam and Youn et al. propose a Quorum-based resource management scheme, which resource Quorum includes middleware entity and also network entity, both can satisfy requirements of application QoS. They suggest a heuristic configuration algorithm in order to optimize performance and usage cost of Resource Quorum [6, 7]. The multiobjective task scheduling problem with multi-QoS constraints is transformed to the general multiobjective combinatorial problem in [8] and an evolutionary algorithm is put forward to solve it. Golconda et al. develop a model with multi-QoS constraints and compares five scheduling heuristics based on that model [9]. Ghosh et al. [10] present scalable QoS optimization algorithms for allocating resources to tasks in a multiprocessor environment. The algorithms are extensions of Q-RAM (QoS-based resource allocation model), can select a QoS operating point, the number of replicas for fault tolerance and the processors on which to run the replicas so as to maximize overall system QoS. All of the above algorithms consider the QoS-based resource scheduling problem from different points of view. In this paper, economic grid QoS-constrained scheduling algorithm is studied.

In this paper, our design is to develop a high-throughput computing scheduling algorithm. We consider scheduling finite resources to satisfy the QoS needs of various grid users. Each task is associated with a benefit function which depends on the finish time of that task, cost of that execution task, and reliability. We build grid resource management, where grid resource users and providers can buy and sell computing resource based on an underlying economic architecture. The agents of the system interact by offering to buy or sell commodities at given prices.

49.3 Grid Resource Management System Model

49.3.1 Resource Scheduling Framework

We design a resource scheduling framework, which consists of three levels: user level, scheduler level, and resource level (as shown in Fig. 49.1). The resource level is the underlying grid resource. Each resource gives its computation capacity and information. The user level is the grid user. Each user expresses their requirements in the form of tasks. The scheduler level is the grid resource management and scheduling system. Through resource scheduling mechanism or policy, users have access to various resources.

There are three key players in the resource scheduling framework:

Resource Provider: it is defined as an entity which shares some particular physical resource within the context of a grid. Given the currently available resources in grid system is m, the grid resource collection $R = \{R1, R2 \ldots Rm\}$, where resources $Ri = \{PEs, MIPS, Bandwidth\}$, the elements of the collection represent the number of processor, the performance of the processor performance, and the available network bandwidth.

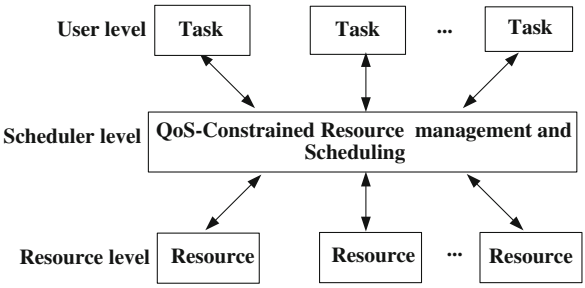

Fig. 49.1 Resource scheduling framework

Resource scheduler: it is responsible for selecting those resources that meet the deadline and budget constraints along with optimization requirements and task assignment. Resource scheduler must make best effort decisions and then submit the tasks to the hosts selected, generally as a user. The scheduler is a mediator between the grid resource user and provider. It can provide the ability to specify resources, obtain quick turnaround for consumer, and receive reliable allocation of resources with all sorts of information gathered by resource discovery.

Resource user: it expresses their requirements in the form of tasks. Each user has a queue of tasks it wishes to execute. Defined a set of scheduled tasks:T = {T1, T2…Tm}, which represents a scheduled task Ti, and Ti = {Length, MIPS, Bandwidth, Deadline}. The meaning of parameter is the length of the grid task, users deadlines demand, the demand for bandwidth, and processing power of CPU MIPS respectively.

49.3.2 QoS Model Requirements

A task executing on a grid system can have a number of QoS requirements that need to be satisfied. It is heterogeneity and dynamics of the grid that make QoS problems in grid environment challenging. Our QoS model is composed of multiple dimensions. Resource selecting is constrained by multiple-QoS metrics such as time, cost, quality, security, reliability, and so on, shown as in the Fig. 49.2.

Time (*T*): it is a common and universal measure of performance. Without loss of generality, the time parameter of a machine is defined as the actual completion time of task running on it. The time can be broken down into three parts that include: transmission time, computation time, and delay time.

Cost (*C*): it represents the cost associated with the execution of grid tasks. Task cost is the cost incurred when a task *t* is executed. Especially, computation resource cost and bandwidth resource cost are ignored. Different resources demands may require different cost. The cost can be broken down into two parts that include: computation cost and bandwidth cost.

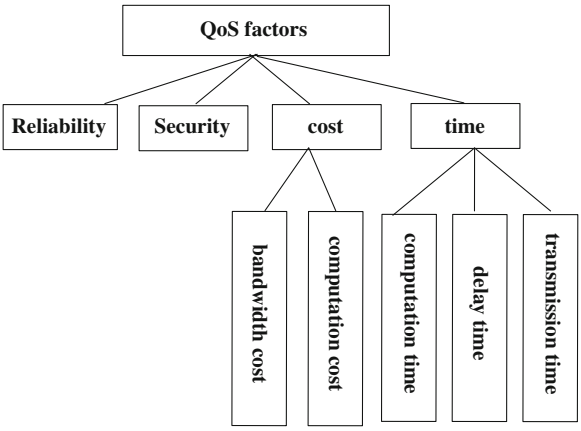

Fig. 49.2 Packet QoS model requirements

Security (*S*): it is defined to be the security level it needs. Each user may require different levels of security for their task and data in a large-scale distributed computing system.

Reliability (*R*): it is defined to be the probability that the task can be completed successfully. A long-time running task may experience failures during its execution, resulting in the wasting of system resources and poor overall performance. Hence, a user may want some reasonable degree of reliability for its task in order to minimize the adverse effects of failures.

Time (*T*), Cost (*C*), Security (*S*), and Reliability (*R*) are considered as the QoS dimensions of a task. As a result, the QoS model of task i can be formulated as Qi = [*T*, *C*, *S*, *R*].

49.4 Grid Task Scheduling Algorithm

From the user point of view, we propose a scheduling strategy for the scheduler.

The expected time *ET* (*Ti*, *Rj*) of task *Ti* on resource *Rj* is defined as the amount of time taken by *Rj* to execute *Ti* given that *Rj* has no load when *Ti* is assigned. The expected time cannot be given directly to the quantitative value of their specific needs calculated by the corresponding value. Therefore, it is defined as follows:

1. ET (Ti, Rj) = Computation time + transmission time + delay time.
2. Computation time = Computational tasks/CPU speed.
3. Transmission time = Data Size/network bandwidth.

The Cost *C* (Ti, Rj) of task Ti on resource Rj is defined as the amount of cost taken by Rj to execute Ti. The Cost *C* (ti, *e*) can be estimated as follows:

C (Ti, Rj) = computation cost + bandwidth cost

The Reliability (Ti) of task Ti is defined to be the probability that the task Ti can be completed successfully.

The Security (Ti) of task Ti is defined to be the security level for the task Ti.

The Cost F of task Ti is an upper bound on the total service cost paid by user.

The uTi is an upper bound on the total completion time.

The scheduling algorithm is considering the QoS constraints when send tasks. Given Q (Ti) = $a \times$ (uTi – ET (Ti, Rj)) + $b \times$ ($F - C$ (Ti, Rj)) + $c \times$ Reliability (Ti) + $d \times$ Security (Ti), where $a + b + c + d = 1, 0 \leq a, b, c, d \leq 1$, a, b, c, d denote the weight assigned to expected time, cost and reliability, security, respectively.

The weight a, b, c, and d indicate that the task requirements degree of four constraints. Scheduling algorithm takes into account the different tasks which demand different resources in the implementation of grid system. According to the demands of the user tasks, the weight a, b, c, and d are set.

While a task with no QoS request can be executed on both high QoS and low QoS resources, a task that requests a high QoS service can only be executed on a resource providing high quality of service. Thus, it is possible for low QoS tasks to occupy high QoS resources while high QoS tasks wait as low QoS resources remain idle. To overcome this shortcoming, we propose the Scheduling algorithm to take the QoS matching into consideration while scheduling.

Step1: {for each task Ti in T = {T1, T2,…, Tm}

{for each resource Rj in R = {R1, R2,…, Rm}
Q (Ti, Rj) = $a \times$ (uTi – ET (Ti, Rj)) + $b \times$ (F–C (Ti, Rj)) + $c \times$ Reliability (Ti) + $d \times$ Security (Ti)
endfor}
endfor}

Step2: {Sort resources by the increasing order of Q (Ti, Rj)}
Step3: {Do until all tasks in T are scheduled

{for each task Ti in T find the highest Q (Ti, Rj) and the corresponding resource
find the task Ti with the maximum highest Q (Ti, Rj)
assign task Ti to Rj that gives it the maximum highest Q (Ti, Rj)
delete Ti from T}
endfor}
enddo}

49.5 Performance Studies

Simulation studies are carried out to evaluate the performance of QoS-Constrained resource scheduling algorithm. We devise other two resource scheduling schemes to compare with QoS-Constrained resource scheduling algorithm in terms of task

completion time. The algorithm one allocates better CPU capabilities than network capabilities to the grid task. The algorithm two randomly allocates network capacity and computation capacity satisfying the minimum QoS requirements to grid task, the new algorithm uses our algorithm. Compared with other two task scheduling algorithm, task completion time our algorithm is shown in Fig. 49.3.

In order to evaluate the performance of QoS-Constrained grid resource scheduling algorithm, the Grid Sim simulation tool is used. This toolkit provides basic functions for the simulation of distributed applications in grid environments. In our experimental testing, we change the number of tasks in the Grid system from 100 to 700. The bandwidths of all links are uniformly distributed between 1 and 200 Mbps. Processor capacity varies from 10 to 100 per time unit, each node' computing delay varies from 1 to 20 per time unit. To characterize various grid resource usage, the simulation abstract both time and resource usage. The resource cost can be expressed in grid dollar that can be defined as processing cost per MIPS.

Figure 49.3 shows that the new QoS-Constrained grid resource scheduling algorithm performs better than the algorithms one and two in terms of the completion time. From the results in Fig. 49.3, for both three algorithms, smaller grid size leads to faster completion times. Our algorithm considers both optimal network resource and CPU allocation, so it has better completion times.

QoS scheduling algorithm adopts both application-centric and resource-centric scheduling objective function aim to optimize the performance of each individual grid user and the performance of the grid resources. It cannot only satisfy the diverse requirements of QoS with different preference from the user perspectives, but also improve the resource utilization rate from the system perspectives. The algorithm is compared with others based on the quality of the prediction formulated by inaccurate information.

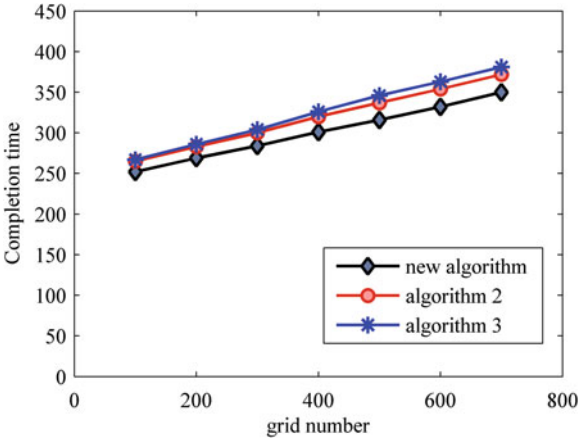

Fig. 49.3 Comparison of completion time

49.6 Conclusion

This study is motivated from the fact that while users with diverse QoS requirements will be having access to a grid computing system, a QoS-Constrained scheduling algorithm is needed not only for meeting users' QoS requirements but also for utilizing the system resources more efficiently. Through the experiments, we have verified the advantages of the algorithm. And it shows that the algorithm can find the most suitable resources and improve resource discovery efficiency. It involves less overhead and leads to better completion times than the other two algorithms.

References

1. He XS, Sun XH, von Laszewski G (2003) QoS guided min–min heuristic for grid task scheduling. J Comput Sci Technol 18(4):442–451
2. Dŏgan A, Özgüner F (2002) Scheduling independent tasks with QoS requirements in grid computing with time-varying resource prices. In: proceedings of GRID 2002, lecture notes in computer science, 2536:58–69
3. Ding D, Luo SW, Gao Z (2007) An object-adjustable heuristic scheduling strategy in grid environments. J Comput Res Dev 44(9):1572–1578
4. Lee C, Lehoczky J, Siewiorek J, Rajkumar R, Hansen J (1999) A scalable solution to the multi-resource QoS problem. In: twentieth IEEE real-time systems symposium 363:25–27
5. Li CL, Li LY (2006) A distributed multiple dimensional QoS constrained resource scheduling optimization policy in computational grid. J Comput Syst Sci 72(4):706–726
6. Nam DS, Youn CH (2004) QoS-constrained resource allocation for a grid-based multiple source electrocardiogram application. In: proceedings of computational science and its applications—ICCSA 2004, lecture notes in computer science 3043:352–359
7. Nama H, Chiang M, Mandayam N (2006) Utility lifetime tradeoff in self regulating wireless sensor networks a cross-layer design approach. In: proceedings of IEEE ICC, pp 3511–3516
8. Zhang WZ, Hu MZ, Zhang H (2006) A multi objective evolutionary algorithm for grid job scheduling of multi-QoS constraints. J Comput Res Dev 43(11):1855–1862
9. Golconda KS, Dogan A (2004) A comparison of static QoS-based scheduling heuristics for a meta-task with multiple QoS dimensions in heterogeneous computing. In: IPDPS 2004 proceedings of the 18th international parallel and distributed processing symposium. IEEE Computer Society Press, Los Alamitos 25:725–741
10. Ghosh S, Rajkumar R, Hansen J, Lehoczky J (2003) Scalable resource allocation for multi-processor QoS optimization. In proceedings of 23rd international conference on distributed computing systems 11:174–183