Secure Requirement Prioritized Grid Scheduling Model

Manjot Kaur Bhatia[1], S. K. Muttoo[2], and M. P. S. Bhatia[3]
(Corresponding author: Manjot Kaur Bhatia)

Research Scholar, Dept. of Computer Science, University of Delhi, Delhi[1]
T Reader, Department of Computer Science, University of Delhi, Delhi[2]
Professor, Dept. of Computer Engineering, Netaji Subhas Institute of Technology, Azad Hind Fauj Marg[3]
Sector 3, Dwaraka, New Delhi-110078
(Email: manjot_bhatia@hotmail.com, skmuttoo@cs.du.ac.in, bhatia.mps@gmail.com)

(Received Oct. 17, 2011; revised and accepted May 5, 2012)

Abstract

Grid computing is a high performance computing environment to solve larger scale computational demands. Grid computing contains resource management, task scheduling, security problems, information management and so on. Task scheduling in an important aspect of distributed computing. As grid computing is a form of distributed computing with heterogeneous resources working in a shared environment with no central control. The main aim of Grid scheduling is to increase the system throughput and to satisfy the job requirements from the available resources. This work proposes a secure requirement (SRP) prioritized task-scheduling algorithm for grid computing. This algorithm is based on scheduling the jobs based on the resource requirement of the jobs which considers the memory requirement as the resource requirement of the jobs. It is named as the secure requirement prioritized (SRP) scheduling algorithm as the jobs memory requirement is passed to scheduler in encrypted form. It is compared with one of the widely used grid scheduling algorithm MinMin and has been tested in simulated grid environment. The experimental results showed a significant improvement in terms of makespan and system utilization.

Keywords: Distributed environment, grid computing, requirement based scheduling, secure grid scheduling, task scheduling

1 Introduction

Grid computing is a form of distributed computing in which resources are geographically distributed and owned by different individuals with different technologies. This distributed environment allows sharing of geographically distributed heterogeneous computers and resources. Users can access and utilize the resources of multiple domains participating in the grid network. It's a new technology that allows easier access to remote computational resources to tackles complex computations. Grid computing aims to maximize the utilization of an organization's computing resources by making them shareable across applications and, potentially, provide computing on demand to third parties as a utility service. It schedule the independent jobs submitted by different users on dynamically distributed resources that increases the overall throughput and also utilizes the unused processors. These resources can be shared by various applications depending upon their availability and QOS requirement of the applications. As grid computing allows user to access remote resources and provide cooperative distributed computing environment, so user jobs can be executed either on local or remote computer systems. The job of the grid scheduler is to automatically assign the suitable resources to the independent jobs to maximize the system utilization. It also reduces the average response time of the jobs. The efficient utilization of grid computing resources can improve the overall job-throughput due to load balancing of the tasks between the grid resources.

The grid scheduling is divided into three phases [10]. These are Resource exploring, Machine selection and Executing. The first phase recovers all the available resources, the second phase deals with finding the best match between the set of jobs and available resources. The phase two is a NP-hard Problem [20]. The behavior of computational grid is dynamic and unpredictable as it depends upon various factors:

1) Network connection;
2) Availability & non-availability of resources at the required time;
3) The number of resources joining & leaving the grid;
4) Performance of grid resources can vary from time to time.

The jobs will take different execution time on different machines. So, task scheduling as a part of grid scheduling

is a problem to schedule a stream of applications from different users to a set of computing resources to minimize the total completion time. This scheduling requires the matching of different jobs with the machines that satisfy their resource requirement. There are two different goals for task scheduling: (i) Increasing computing performance, its aim is to minimize the execution time of each application that is considered in parallel processing. (ii) Increasing overall throughput, its purpose is to schedule a set of independent tasks in such a way that it increases the processing capacity of the systems for long period of time.

We have focused on the second goal and propose a new task-scheduling algorithm for grid computing that provides high throughput and efficient utilization of resources. This grid-scheduling algorithm tries to minimize the total turnaround time of the jobs. The resources in grid environment are not only dedicated to grid applications, as they have to handle their own local jobs also. So, the grid jobs need to compete for the resources according to their resource requirement. In this scheduling algorithm we are considering the main-memory as the resource requirement of the grid jobs. The scheduler considers the resource requirement of the grid jobs and assigns the jobs to the resources that satisfy their resource requirement. To secure the requirements of the tasks, memory requirement of the tasks is encrypted using RC5 algorithm and then passed to the scheduler. Our Scheduling algorithm increases the efficiency and utilization of grid resources by scheduling the jobs based on their resource requirement and provides secure communication with scheduler.

2 Literature Survey

In the past few years, researchers have proposed scheduling algorithms for parallel system [4, 20, 22]. However, the problem of grid scheduling is still more complex than the proposed solutions. Therefore, large number of researchers [2, 6, 12, 16, 17] is showing interest in it. Current systems of grid resource management was surveyed and analyzed based on classification of scheduler organization, system status, scheduling and rescheduling policies [11]. However, the characteristics and various techniques of the existing grid scheduling algorithms are still complex particularly with extra-added components.

Job scheduling on grid computing not only aims to find an optimal resource to improve the overall system performance but also to utilize the existing resources more efficiently. Recently, many researchers have studied the problem of job scheduling algorithm on grid environment. Some of those are the popular heuristic algorithms, which have been developed, are UDA, OLB, min-min, the fast greedy, GA, SA, tabu search [1] and an Ant System [14]. These algorithms have several advantages and have some drawbacks also. UDA assign too many jobs to a single grid node. This leads to overloading and the increases the response time of the jobs. The drawback of OLB is that it does not achieve the load balance and leads to hard calculation of minimum completion time for a job. The algorithms GA, SA and GSA are difficult to implement. The heuristic algorithms proposed for job scheduling in [1] and [14] depend on static environment of system load and the expected value of execution times. Li [13] proposed a scheduling algorithm in which job will be moved from one machine to another machine, so the traffic in the grid system will be automatically increased. Yan Hui [7] has taken into account communication cost and different ant agents.

Currently available Grid Resource Management system like: Condor, Globus, NetSolve, Nimrod/G, AppLeS uses different Grid scheduling approaches. The Condor uses centralized scheduler and designed to improve overall throughput of the system in a controlled network environment. Its scheduling algorithm does not consider any QoS requirement of jobs. The AppLeS scheduling algorithm focuses on efficient co-location of data and experiments as well as adaptive scheduling. The Nimrod uses decentralized scheduler and its scheduling approach is based on predictive pricing model and Grid economy model. The Netsolve has decentralized scheduler and scheduling approach focuses on fixed application oriented policy considering soft QoS. In our algorithm we consider QoS in scheduling. Our proposed algorithm is different from the above given algorithms; it considers the memory requirement of the job and assigns the jobs to the available resource accordingly. The above discussed algorithms are not security aware and security is an important concern in Grid Scheduling, considering security constraint modifies the schedule of the scheduling algorithms. The secure grid scheduler should meet the security requirements of the jobs and also tries to minimize the makespan, average response time of the jobs by utilizing the resources effectively.

Secure Grid Scheduling: In Grid computing resources and data from different administrative domains work together as virtual organization. Computations of tasks at different resources of different administrative domains give rise to security issues in Grid scheduling. Various researchers have proposed and developed secure grid scheduling model considering different security requirements of Grid Scheduling. Ian Foster [5] proposed a security architecture that addresses requirements like user, resource, process authentication with each other and dynamically varying resource requirements of the processes. Wu and Sun [20] proposed a genetic algorithm addressing heterogeneities of fault tolerant mechanism in computational grid. Song [19] considers risk involved in dispatching the jobs to remote nodes and proposed three secure scheduling algorithms on different risk levels. Kołodziej [9] formalizes the Grid scheduling problem as a non-cooperative non-zero sum game of the Grid users in order to address the security requirements. He considered users' cost of playing the game as a total cost of the secure job execution in Grid and tries to minimize the total cost by using four genetic-based hybrid meta-heuristics. Kashyap [8], proposed a security-aware Grid scheduling model, in this paper author quantifies, estimates security overhead and considers security requirement as the main concern for

scheduling tasks on the grid nodes. She incorporated this idea of prioritizing security requirements on the existing scheduling heuristics Min-Min and Max-Min. Xie and Qin [21] proposed a scheduling algorithm SAHA, schedules data intensive jobs considering their security requirements on data grids. He introduced a concept of security deficiency and also proposed strategy to enhance security of jobs.

3 Proposed Model

3.1 Proposed Model Architecture

In this model, we assume a computing grid composed of a number of independent non-dedicated sites with several heterogeneous computational resources of various organizations. As the sites are non-dedicated, no one has full control on all the available resources and applications (jobs). Each site has local users that submit jobs to its own local job scheduler and the local job scheduler is responsible for managing local jobs only. For this reason, the job scheduling in this environment is complicated.

The independent users submit their jobs to the Grid Scheduler. A grid job consists of n independent tasks. Each task is characterized by file size and encrypted memory requirement. The aim of this model is find the optimal schedule for assigning the jobs to the processing nodes that satisfies the memory requirement by providing the memory requirement in encrypted form so that the intruders should not be able to modify the requirement of the jobs and also give minimum makespan. The memory requirement of the jobs is encrypted using the RC5 algorithm. RC5 is symmetric block cipher that uses the same cryptographic key for encryption and decryption. It has the variable length cryptographic key depending upon the level of security. The scheduler retrieves the information of the grid resources such as processing speed, memory capacity. The data retrieved from the participating sites is used to find the optimal resources according to job resource requirement for processing of jobs. The scheduler decrypts the memory requirement of the tasks and schedules the tasks depending upon its requirement.

3.2 Terminologies Used

The major objective of our algorithm is to allocate the best suitable machine to the tasks, arriving at the Grid Scheduler. As our task scheduling algorithm is based on the availability of memory required by the task.

Here is the list of terminologies and its definitions, used in this paper:

J_{list} is the list of all the task of the given job that is to be scheduled. A task is characterized as T_i ($size_i$, mr_i) where, $size_i$ is the size of the task and mr_i is the encrypted memory requirement of the task. A processing node is characterized as P_j (ps_j, mc_j, bt_j) where, ps_j is the processing speed of the node, mc_j is the maximum memory capacity of the processing node and bt_j is the begin time at j^{th} node. Tw is the waiting time of the node (time to execute tasks already assigned to the node). $P_{rs,i}$ is the list of nodes satisfying the memory requirement of i^{th} task i.e. on which the i^{th} task can be executed.

The time spent by the i^{th} process waiting for the j^{th} node (Grid Scheduler) is

$$Tw_{ij} = \text{Max}(\min \boldsymbol{tp_j}(\text{avail}),\ \min \boldsymbol{tme_j}(\text{avail})),$$

where $\boldsymbol{tp_j}$ is time that shows after how much time the i^{th} task has got the j^{th} node, $\boldsymbol{tme_j}$ is time i.e after how long time the i^{th} task has got the j^{th} node that satisfy its memory requirement($\boldsymbol{me_i}$). Let $\boldsymbol{ET_{ij}}$ is defined as the amount of time taken by processing node P_j, to execute task t_i, given that P_j has no load when task T_i is assigned.

$$ET_{ij} = outTime_{ij} - inTime_{ij}$$

Where $\boldsymbol{outTime_{ij}}$ is the at which i^{th} process is completed in the jth processor. $\boldsymbol{inTime_{ij}}$ is the time at which i^{th} process is submitted to the j^{th} processor.

The completion time C_{ij} of the i^{th} process at j^{th} machine is:

$$C_{ij} = Tw_{ij} + ET_{ij.}$$

3.3 Scheduling Algorithm

Make span is a measure of the throughput of the heterogeneous system. The aim of our grid scheduling algorithm is to minimize the makespan. The heuristic can be divided into two categories online mode and the other batch mode. In online mode, whenever a job arrives to the scheduler it is allocated to the first free machine. In this method, the arrival order of the job is important. Each task is considered only once for matching and scheduling. In case of batch mode, the jobs are collected in a set and are examined for mapping at prescheduled times called mapping events. This independent event uses heuristic approach to make better decision. This mapping heuristics do better task/host mapping because the heuristics have the resource requirement information for the meta-task, and know the actual execution time of a larger number of tasks. Several heuristics approaches like Min-min, Max-min, UDA and GA are proposed for scheduling independent tasks. Most of these algorithms consider the expected execution time of each task as the criteria to make better decision. The general scheduling algorithms does not consider the resource requirement, which affects scheduling process in a Grid. Regardless of their computing power request, some tasks may require more memory whereas others can be satisfied with less memory. For e.g. If scheduler assigns a task that require less memory for execution on the processor with high memory, tasks requiring high memory will then have to wait. Considering memory as the resource requirement in scheduling should lead to a better scheduling algorithm. Based on this requirement, a new scheduling algorithm considering 'memory as resource requirement' is proposed. It works as follows:

- Sort and make a list of tasks based on their memory requirement as (T_{sort}) from complete list of tasks $T_{list.}$
- To avoid attack-in-the-middle encrypt the memory requirement of the tasks in the T_{sort} list using RC5 algorithm as the maximum size would not exceed 16 bytes and make a list of tasks with their encrypted memory requirement $ET_{sort.}$
- Submit the list ET_{sort} to the scheduler.
- After receiveing the ET_{sort} list the scheduler decrypt the memory requirement of the tasks and create DT_{sort}.
- For each task of updated DT_{sort}, find the list of the nodes ($N_{satisfy}$) which satisfy the memory requirement of the task.
  Compute the computation time for each task of DT_{sort} on its entire node list $N_{satisfy.}$
- For each task**,** mark the node from the $N_{satisfy}$ that gives minimum completion time.
  - For all such task-node pair, allocate the task to the respective marked node.
  - Remove the task from the DT_{sort} list.
  - Modify the waiting time of the resource.
  - Repeat the entire process till DT_{sort} list is empty.
  - After all the tasks from the DT_{sort} are allocated new T_{sort} is created and the entire process begins again.

Algorithm 1: Scheduling algorithm

1. for all tasks T_i
2. create T_{sort} from T_{list}
3. create ET_{sort} from T_{sort}
4. end for
5. for all tasks in ET_{sort}
6. create DT_{sort}
7. endfor
8. do until (DT_{sort} != NULL)
9. {
10. for each task t_i in the DT_{sort}
11. **{**
12. create $N_{satisfy,i}$
13. for each node j from the node list $N_{satisfy,i}$
14. compute
15. Tw_{ij}=Max(min tp_j (avail), min tme_j (avail))
16. $C_{ij} = Tw_{ij} + ET_{ij}$
17. find the completion time for each task and its
18. corresponding node.
19. Generate matrix $CT_{i,j}$
20. }
21. From the matrix CT, find the task with
22. Minimum CT=($CT_{i,j}$)
23. Schedule task i on node j
24. Delete task t_i from DT_{sort} and T_{list}
25. Modify $Tw_{ij} = Tw_{ij} + CT_{ij}$ for the j^{th} node
26. }

4 Experimental Testing

To evaluate the performance of newly proposed SRP scheduling algorithm in Grid environment and to compare it with existing algorithm Min-Min, a simulator is designed in Java. The simulator consists of Nodes, Tasks and SRP algorithm for generating heterogeneous grid nodes and tasks sets randomly within the specified range. In the experimental testing we used heterogeneous machines with different processing speed, memory capacity and tested it for different number of tasks (e.g. size of tasks, memory requirement), shown in Table 1.

Table 1: Parameters for the simulation experiments

Parameters	Values
No. of nodes	4
Processing speed of nodes(ps)	1-10 (MIPS)
Memory capacity of processing nodes	200-350 (MB)
No. of tasks	10-20
Size of Tasks	10-200 (MB)
Memory requirements	150-300 (MB)

We compared the results of our SRP scheduling algorithm with the most widely used min-min algorithm.

We compared the makespan of our algorithm with Min-Min on heterogeneous environment by varying the processing speed and memory capacity of the Grid nodes. The experiments are conducted on different number of tasks ranging from 10 to 20. It is observed that makespan for the given number of tasks is either shorter or equal in case of SRP based scheduling algorithm as compare to Min-Min algorithm. Our algorithm has shown an improvement over Min-Min algorithm. The results are shown in Table 2 and Figure 1.

Table 2: Makespan(in secs) for SRP and Min-Min

No. of tasks	Min-min	SRP	Improvement
12	74	65	12.1 %
16	70	60	14%
18	64	54	16.5 %

5 Conclusion and Future Work

In this paper we have proposed a scheduling algorithm that considers the resource requirement of the jobs the Grid environment. To secure the resource requirement of the jobs so that it should not be modified by any intruder or any other task, we are sending this information to the scheduler in the form of encrypted data. The scheduler on the other end decrypts the task's memory requirement and schedules the tasks accordingly.

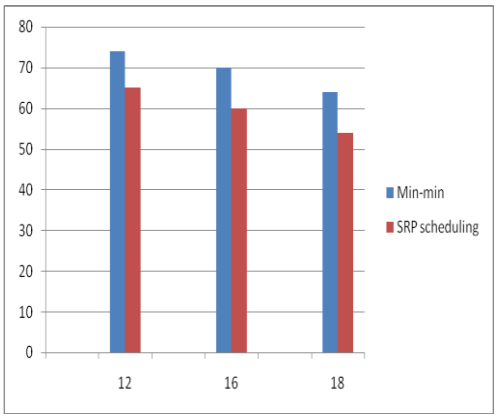

Figure 1: Makespan of min-min and SRP based scheduling

This newly proposed scheduling algorithm achieve high throughput in the Grid computing. A simulation system was developed to test the Secure Requirement Prioritized scheduling algorithm in a simulated Grid environment. We used the makespan time of batch jobs as the comparison criteria. When Compared with Min-Min, the experimental results show that SRP scheduling algorithm show a noticeable increase in performance and provide security in information exchange between jobs and scheduler as compared to MIN-MIN algorithm. As memory is an important resource, this research work considers memory and securing resource requirement as an important factor of the job. In future, research can be done on the Secure Grid Scheduling of jobs by considering and securing multiple requirements of the jobs.

References

[1] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810-837, 2001.

[2] E. Carsten, V. Hamscher, and R. Yahyapour, "Economic scheduling in grid computing," in *Proceedings of 8th International Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 128-152, Springer-Verlag London, UK, 2002.

[3] D. Fernandez-Beca, "Allocating modules to processors in a distributed System," *IEEE transaction on Software Engineering*, pp. 1427-1436, 1989.

[4] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong. "Theory and practice in parallel job scheduling," in *3rd Workshop on Job Scheduling Strategies for Parallel Processing*, LNCS 1291, pp. 1-34, 1997.

[5] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A security architecture for computational grids," in *ACM Conference on Computer and Communications Security*, pp. 83-92, 1998.

[6] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour, "Evaluation of job-scheduling strategies for grid computing," in *Proceedings of First IEEE/ACM International Workshop on Grid Computing*, LNCS 1971, pp. 191-202, Springer-Verlag, Berlin, 2000.

[7] Y. Hui, X. Q. Shen, X. Li, and M. H. Mu, "An Improved ant algorithm for Job Scheduling in Grid Computing," in *IEEE Fourth International Conference on Machine Learning and Cybernetics*, pp. 18-21, Guangzhou, Aug. 2005.

[8] R. Kashyap and D. P. Vidyarthi, "Security-aware scheduling model for computational grid," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 2, pp. 1377-1391, 2011.

[9] J. Kołodziej and F. Xhafa, "Meeting security and user behavior requirements in Grid scheduling," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 213-226, 2011.

[10] J. Krallmann, U. Schwiegelshohn, and R. Yahyapour, "On the design and evaluation of job scheduling algorithms," in *5th Workshop on Job Scheduling Strategies for Parallel Processing*, LNCS 1659, pp. 17-42, Spriger-Verlag, 1999.

[11] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of Grid resource management systems for distributed computing," *Software: Practice. Experince*, vol. 32, pp. 135–164, 2002.

[12] K. Li, "Job scheduling and processor allocation for grid computing on metacomputers," *Journal of Parallel and Distributed Computing*, pp. 1406-1418, 2005.

[13] L. Li, Y. Yang, L. Li and W. Shi, "Using ant optimization for super scheduling in computational grid," in *IEEE proceedings of the 2006 IEEE Asia-pasific Conference on Services Computing*, 2006.

[14] G. Ritchie and J. Levine, "A fast, effective local search for scheduling independent jobs in heterogeneous computing environments," in *Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group*, pp. 178-183, 2003.

[15] R. L. Ronald, "The RC5 encryption algorithm," in *Proceedings of the 1994 Leuven Workshop on Fast Software Encryption*, pp. 86-96, 1995.

[16] J. M. Schopf, "A general architecture for scheduling on the grid," *Special issue of JPDC on Grid Computing*, Apr. 2002.

[17] H. X. Shan, X. H. Sun, and G. V. Laszewski, "A QoS guided scheduling algorithm for grid computing," in *proceddings of Int. Workshop on Grid and Cooperative Computing*, pp.745-758, Sanya, China, 2002.

[18] G. C. Sih and E. A. Lee, "A compile-time scheduling heuristic for interconnection constrained heterogeneous

processor architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, pp. 175-187, Feb.1993.

[19] S. Song, K. Hwang, and Y. K. Kwok, "Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling," *IEEE Transactions on Computer*; vol. 55, no. 6, pp. 703-719, 2006.

[20] C. C. Wu and R. Y. Sun, "An integrated security-aware scheduling strategy for large-scale computational grid," *Future Generation Computer Systems*, vol. 26, no. 2, pp. 198-206, 2010.

[21] T. Xie and Q. Xiao, "SAHA: A Scheduling algorithm for security-sensitive jobs for data grids," in *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, pp. 22, May 2006.

[22] B. Zhou and X. Qu "An efficient scheduling algorithm for multiprogramming on parallel computing system," in *Proceedings of the 20th Australasian Computer Science Conference*, pp. 336-345, Sydney, Australia, Feb. 1997.

Manjot Bhatia: She is a Research Scholar in Department of Computer Science at University of Delhi, Delhi,India. She is MPhil, MCA and pursuing her PhD (Computer Science). She has more than ten years of teaching experience in the areas of Operating system, Grid computing, Linux atc. Her research areas include "Grid Computing and Security" on which papers have been published in International conferences and journals. Various seminars, workshops and FDP (AICTE) have been attended.

Sunil Kumar Muttoo: He is working as an Associate Professor in Department of Computer Science at University of Delhi, Delhi, India. He is M.Tech., M.Phil., Ph.D. and from Delhi University. He has more than seventeen years of teaching experience at Post Graduate and Doctoral level. His teaching areas include Computer Security, Computer Graphics, Steganography etc. His research areas include Information Hiding and Coding Theory. He has published more than thirty papers in various National and International conferences and journals.