American Journal of Engineering Research (AJER)
e-ISSN : 2320-0847 p-ISSN : 2320-0936
Volume-02, Issue-05, pp-78-83
www.ajer.us

Research Paper Open Access

Implementing Short Term Scheduler and Preemptive Algorithm in Grid Computing Scheduling

CH.V.T. E.V Laxmi, K.Somasundaram,
1.Research Scholar (Karpagam University), Associate Professor, Department of Computer Science Engineering, Raghu Engineering Collge, Visakhapatnam,Andhra Pradesh,India.
2.Professor, Department of Computer Science and Engineering,Jaya Engineering College, CTH Road, Prakash Nagar,Thiruninravur,Thiruvallur - Dist, Tamilnadu.

Abstract: Grid Computing has emerged as an important new field focusing on resource sharing. One of the most challenging issues in Grid Computing is efficient scheduling of tasks and there is need of finding faster and cheaper solutions to solve computational problems. The deployment of Grid systems involves the efficient management of heterogeneous, geographically distributed and dynamically available resources. However, the effectiveness of a Grid environment is largely dependent on the effectiveness and efficiency of its schedulers, which act as localized resource brokers. This article provides a brief overview on grid computing and Scheduling processes, important factors considered in resource management and Scheduling, comparison of different Scheduling processes and future outlook of its resource management and Scheduling in Grid systems by implementing short term scheduling. This paper investigates the use of short term scheduling mechanism in grid computing and implementing the Shortest Remaining Time First (SRTF) algorithm for the selection of jobs from the job pool.

Keywords: *Grid Computing, Resource Management, Scheduling, Resource Sharing, scheduling processes*

I. INTRODUCTION

The Grid is emerging as a new paradigm for solving problems in science, engineering, industry and commerce. Increasing numbers of applications are utilizing the Grid infrastructure to meet their computational, storage and other needs. A single site can simply no longer meet all the resource needs of today's demanding applications, and using distributed resources can bring many benefits to application users. The deployment of Grid systems involves the efficient management of heterogeneous, geographically distributed and dynamically available resources. However, the effectiveness of a Grid environment is largely dependent on the effectiveness and efficiency of its schedulers, which act as localized resource brokers. Figure 1.1 shows that user tasks, for example, can be submitted via Globus to a range of resource management and job scheduling systems, such as Condor, the Sun Grid Engine (SGE), the Portable Batch System (PBS) and the Load Sharing Facility (LSF). Grid scheduling is defined as the process of mapping Grid jobs to resources over multiple administrative domains. A Grid job can be split into many small tasks. The scheduler has the responsibility of selecting resources and scheduling jobs in such a way that the user and application requirements are met, in terms of overall execution time (throughput) and cost of the resources utilized.

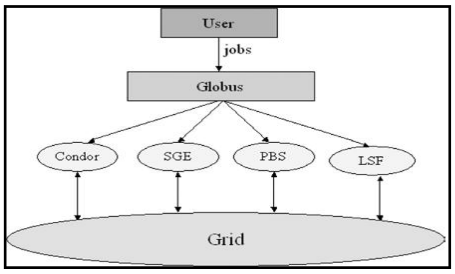


Figure1.1

Jobs, via Globus, can be submitted to systems managed by Condor, SGE, PBS and LSF

II. RELATED WORK

Brief Description of SCHEDULING PARADIGMS

In the grid-computing system, the use of computer resources from multiple administrative domains that are applied collectively to solve a problems that has demanding requirements such as a storage space, bandwidth etc. therefore these resource has to be scheduled, present there are three scheduling paradigms– centralized, hierarchical and distributed.

Centralized scheduling

In a centralized scheduling environment, a central machine (node) acts as a resource manager to schedule jobs to all the surrounding nodes that are part of the environment. This scheduling paradigm is often used in situations like a computing centre where resources have similar characteristics and usage policies. Figure 1.2 shows the architecture of centralized scheduling. One advantage of a centralized scheduling system is that the scheduler may produce better scheduling decisions because it has all necessary, and up-to-date, information about the available resources. However, centralized scheduling obviously does not scale well with the increasing size of the environment that it manages. The scheduler itself may well become a bottleneck, and if there is a problem with the hardware or software of the scheduler's server, i.e. a failure, it presents a single point of failure in the environment.

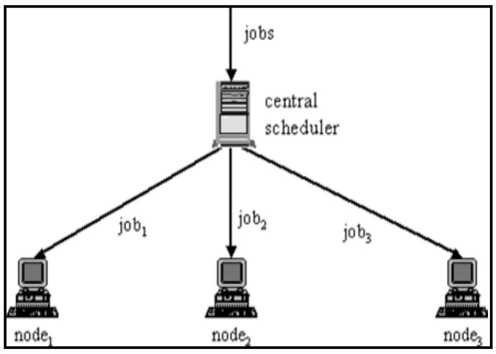


Figure 1.2 Centralized scheduling

Distributed scheduling

In this paradigm, there is no central scheduler responsible for managing all the jobs. Instead, distributed scheduling involves multiple localized schedulers, which interact with each other in order to dispatch jobs to the participating nodes. There are two mechanisms for a scheduler to communicate with other schedulers – direct or indirect communication. Figure 1.3 shows the direct communications in distributed scheduling.

Direct communication

In this scenario, each local scheduler can directly communicate with other schedulers for job dispatching. Each scheduler has a list of remote schedulers that they can interact with, or there may exist a central directory that maintains all the information related to each scheduler. Figure 1.3 shows the architecture of direct communication in the distributed scheduling paradigm.

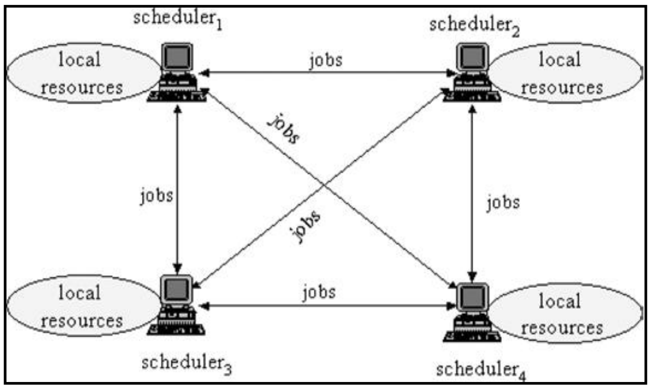


Figure 1.3 Direct communications in distributed scheduling

Communication via a central job pool

In this scenario, jobs that cannot be executed immediately are sent to a central job pool. Compared with direct communication, the local schedulers can potentially choose suitable jobs to schedule on their resources. Policies are required so that all the jobs in the pool are executed at some time. Figure 1.4 shows the architecture of using a job pool for distributed scheduling.

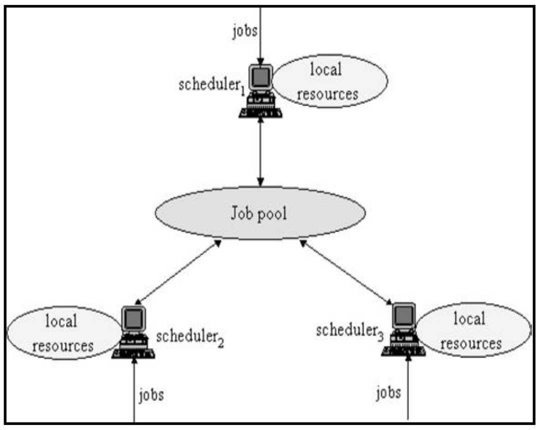


Figure 1.4 *Distributed scheduling with a job pool*

Hierarchical scheduling

In hierarchical scheduling, a centralized scheduler interacts with local schedulers for job submission. The centralized scheduler is a kind of a meta-scheduler that dispatches submitted jobs to local schedulers. Figure

1.4 shows the architecture of this paradigm.

Similar to the centralized scheduling paradigm, hierarchical scheduling can have scalability and communication bottlenecks. However, compared with centralized scheduling, one advantage of hierarchical scheduling is that the global scheduler and local scheduler can have different policies in scheduling jobs.

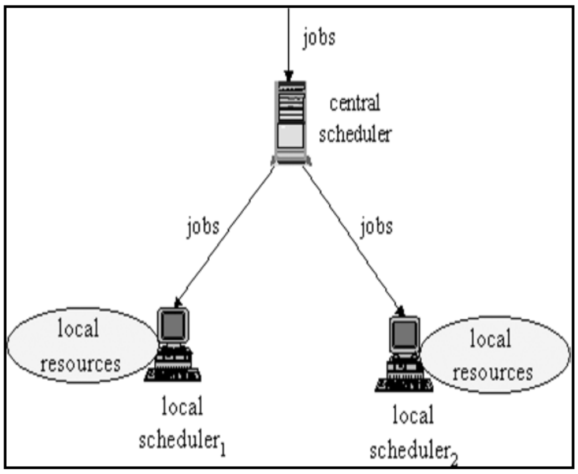


Figure 1.4 Hierarchical scheduling

III. PROPOSED APPROACH

This section explains our proposed approach of new scheduling that is hierarchical scheduling with job pool. In hierarchical scheduling, a centralized scheduler interacts with local schedulers for job submission and then it dispatches submitted jobs to local schedulers. Therefore there is no guarantee of execution of all the submitted jobs which are in the central scheduler. So aim is to execute all the jobs therefore by implementing job pool in between the central scheduler and to the local schedulers, we can give the guarantee of execution of all the jobs which are submitted at the centralized scheduler. Jobs which cannot be executed immediately such jobs are sent to the job pool. The approach aim is, all the jobs should be executed which are submitted to central schedulers by sending the jobs which cannot be executed immediately to the job pool and by implementing special policies on all those jobs which cannot be executed immediately. Figure 1.5 shows the proposed architecture of hierarchical scheduling with job pool.

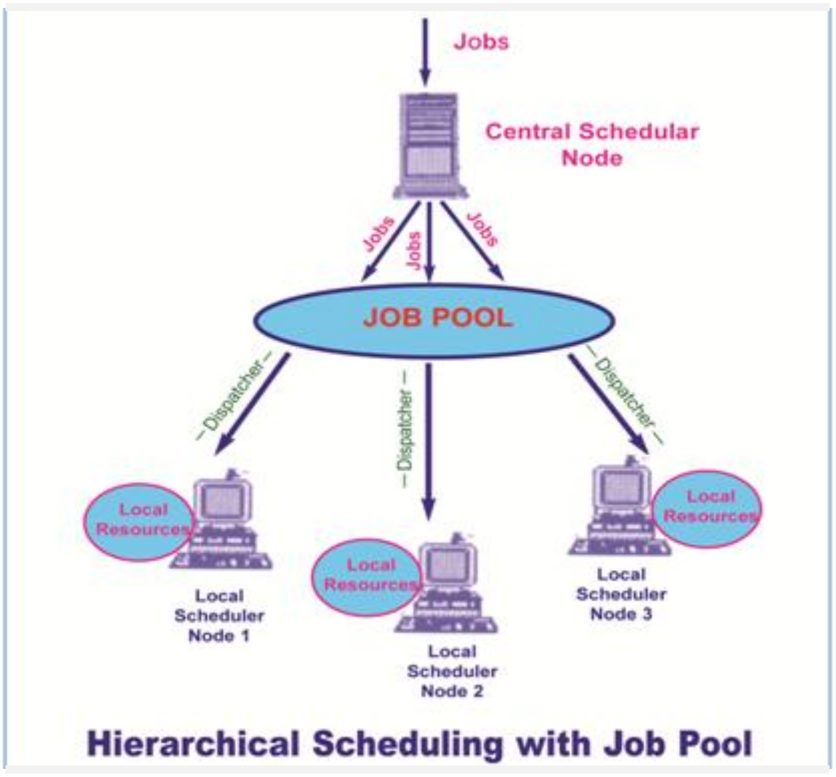


Figure 1.5 Hierarchical scheduling with job pool

3.1 Implementing Short Term Scheduler in Hierarchical scheduling with job pool

The main function of the short term scheduler is, selecting a job from the pool of jobs, which we demonstrated in the figure 1.5, as Hierarchical scheduling with job pool. The short term scheduler gives the control of the CPU of central scheduler with the help of 'Dispatcher'. A dispatcher is a module; it connects the CPU to the process selected by the short term scheduler. The main function of the dispatcher is switching, it means switching the CPU from one process to another process. The method of selecting a job from job pool is depends on the scheduling algorithm. Figure 1.5.1 shows the Queuing diagram for the hierarchical scheduling with job pool.

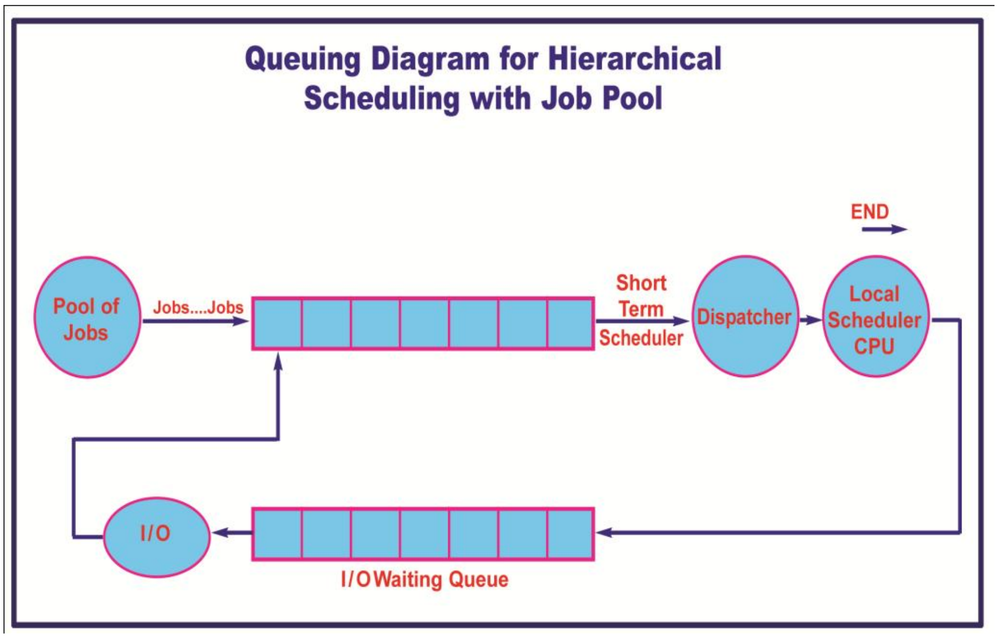


Figure 1.5.1 Queuing diagram for Hierarchical scheduling with job pool

3.2 Shortest Remaining Time First (SRTF) Algorithm

There are many grid scheduling algorithms available. Different grid scheduling algorithms have different properties, in choosing which algorithm to use in a particular situation; we must consider the properties of the various algorithms. Shortest Remaining Time First (SRTF) is the preemptive scheduling algorithm, in which the short term scheduler always chooses the process that has the shortest remaining processing time. When a new process enters, the short term scheduler compare the remaining time of executing process and new process. If the new process has the least CPU burst time, the scheduler selects that job and connects to the CPU of the local nodes; otherwise it continues the old process.

3.3 Case study and experimental Results

The idea of the above algorithm is illustrated considering the case study with the below problem.

JOBS	Local Nodes Burst Time	Arrival Time
J1	3	0
J2	2	6
J3	4	4
J4	6	5
J5	2	8

Figure 1.5.2 shows the arrival chart for the above problem

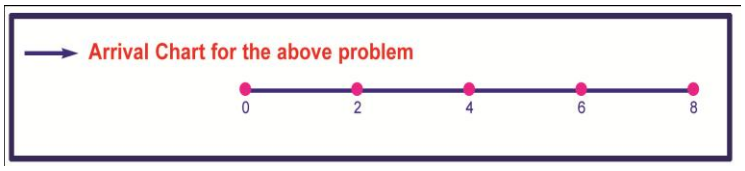
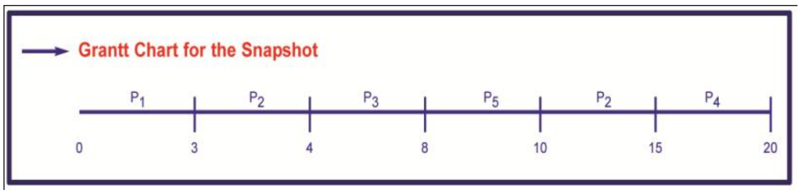


Figure 1.5.3 shows the Gantt chart for the snapshot



Job J1 arrives at time 0, so J1 executing first, J2 arrives at time 2 mi.sec, so compare the J1 remaining time (3-2=1) and J2 time (1<6) so continue the job J1. After the completion of J1, executing the process J2, at time 4 J3 arrives, so compare the reaming time of J2(6-1=5) and burst time of J3(4),(4<5) so CPU of the local scheduler shift to job J3. At time 6 J4 arrives.

Then compare the remaining time for J3(4-2=2) and local node burst time of J4 is 5. (2<5). So continue the process J3. After the completion of J3, the central scheduler consisting of J5, J2, J4. J5 is the least out of three, so executing the J5. After that execute the J2,then next J4.

Now we have to calculate the Average Turnaround Time (ATT), Average Response Time (ART) and Average Relative Delay (ARD).

- Turn Around Time:

Turn around time =	first time-arrival time					
Turn around for J1	=	3	-	0	=	3
Turn around for J2	=	15	-	2	=	13
Turn around for J3	=	8	-	4	=	4
Turn around for J4	=	20	-	6	=	14
Turn around for J5	=	10	-	8	=	2

Average turn around time= 36/5=7.2

- Relative delay or normalized turn around time:

Relative delay= Tq/Ts=turn around time/service time

Relative delay for

J1	=	3/3	=	1.00
J2	=	13/6	=	2.17
J3	=	4/4	=	1.00
J4	=	14/5	=	2.80
J5	=	2/2	=	1.00

Average Relative Delay=1+2.17+1+2.80+1/5

7.97/5

1.59

IV. CONCLUSION

Grid computing is an important tool that is used for both scientific and industrial purposes, which provides an environment with a high amount of resources for computational purposes to solve complex problems. These resources can be scientific instruments, storage devices, network bandwidth, sensors and processors which belong to different proprietary organizations.

This paper carries out a survey on Grid scheduling from the study of different researches carried out on this field. An algorithm of job scheduling has been proposed within this paper. In this paper a brief overview on grid computing and scheduling processes, important factors considered in resource management and scheduling has been proposed. This paper investigated the use of short term scheduling mechanism in grid computing and implementing shortest remaining time first (SRTF) algorithm for the selection of jobs from the job pool has been discussed with a case study.

REFERENCES

[1] Ian Foster and Carl Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Elsevier Inc., Singapore, Second Edition, 2004.

[2] Raksha Sharma, Vishnu Kant Soni, Manoj Kumar Mishra, Prachet Bhuyan A Survey of Job Scheduling and Resource Management in Grid Computing World Academy of Science, Engineering and Technology 40 2010

[3] Edi.Laxmi Scheduling in Grid Computing International Journal of Computer Science and Management Research Vol 2 Issue 1 January 2013

[4] Hamscher, V., Schwiegelshohn, U., Streit, A. and Yahyapour, R. Evaluation of Job-Scheduling Strategies for Grid Computing. GRID 2000, 191–202, 17–20 December 2000, Bangalore, India. Lecture Notes in Computer Science, Springer-Verlag.

[5] Srinivasan, S., Kettimuthu, R., Subramani, V. and Sadayappan, P. Characterization of Backfilling Strategies for Parallel Job Scheduling. ICPP Workshops 2002, 514–522, August 2002, Vancouver, BC, Canada. CS Press. DAGManager, http://www.cs.wisc.edu/condor/dagman/.

[6] Ghare, G. and Leutenegger, S. Improving Small Job Response Time for Opportunistic Scheduling. Proceedings of 8th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2000), San Francisco, CA, USA. CS Press.

[7] Raman, R., Livny, M. and Solomon, M. Matchmaking: Distributed Resource Management for High Throughput Computing. Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing, July 1998, Chicago, IL, USA. CS Press.

[8] Enterprise Edition policy, http://www.sun.com/blueprints/0703/817-3179. pdf.

[9] N1GE 6 Scheduling, http://docs.sun.com/app/docs/doc/817-5678/ 6ml4alis7?a=view. PBS Pro, http://www.pbspro.com/.

[10] Figueira, M., Hayes, J., Obertelli, G., Schopf, J., Shao, G., Smallen, S. ,Spring, N., Su, A. and Zagorodnov, D. Adaptive Computing on the Grid Using AppLeS. IEEE Transactions on Parallel and Distributed Systems, 14(4): 369–382 (2003). NWS, http://nws.cs.ucsb.edu/.

[11] Dail, H., Berman, F. and Casanova, H. A Decoupled Scheduling Approach for Grid Application Development Environments. Journal of Parallel Distributed Computing, 63(5): 505–524 (2003).

[12] Abramson, D., Giddy, J. and Kotler, L. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? Proceedings of the International Parallel and Distributed Processing (IPDPS 2000), May 2000, Cancun, Mexico. CS Press.

[13] Gerasoulis, A. and Jiao, J. Rescheduling Support for Mapping Dynamic Scientific Computation onto Distributed Memory Multiprocessors. Proceedings of the Euro-Pa '97, August 1997, Passau, Germany. Lecture Notes in Computer Science, Springer-Verlag.

[14] Goux, Jean-Pierre, Kulkarni, Sanjeev, Yoder, Michael and Linderoth, Jeff. Master-Worker: An Enabling Framework for Applications on the Computational Grid. Cluster Computing, 4(1): 63–70 (2001).

[15] Cactus, http://www.cactuscode.org/. **300** GRID SCHEDULING AND RESOURCE MANAGEMENT

[16] Spooner, D., Jarvis, S., Cao, J., Saini, S. and Nudd, G. Local Grid Scheduling Techniques using Performance Prediction, IEE Proc. – Comp. Digit. Tech., 150(2): 87–96 (2003).

[17] Young, L., McGough, S., Newhouse, S. and Darlington, J. Scheduling Architecture and Algorithms within the ICENI Grid Middleware. Proceedings of the UK e-Science All Hands Meeting, September 2003, Nottingham, UK.

[18] YarKhan, A. and Dongarra, J. Experiments with Scheduling Using Simulated Annealing in a Grid Environment. Proceedings of the 3rd International Workshop on Grid Computing (GRID 2002), November 2002, Baltimore, MD, USA. CS Press.