Volume 4, No. 3, March 2013 (Special Issue)

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Trust Based Job Scheduling Algorithm in Grid Network

Kiran Rayapu
Department of Computer and Information Sciences
University of Hyderabad Hyderabad, India
rkiran@idrbt.ac.in

Shakti Mishra
Institute for Development and Research in Banking Technology Hyderabad, India
smishra@idrbt.ac.in

Abstract: The Computational grid provides promising platform for efficient execution of resource intensive parameter sweep applications or jobs. But the parameter sweep jobs may or may not execute within the estimated execution time or estimated resource requirements. Sometimes the jobs can hold the resources longer time than the estimated time. It increases the turnaround time of other jobs that always execute within the estimated execution time. This imposes a challenge of scheduling grid jobs in an efficient way so that neither of the jobs starves. Although numerous works have been done in grid scheduling, no grid scheduling algorithm has considered all the factors like trust, resource need, priority and time consumption. To address this problem, a trust based job scheduling algorithm has been proposed and developed. The proposed approach considers all the factors aforementioned to calculate Scheduling factor of a job and the jobs can be scheduled on the basis of Scheduling factor. An empirical study has also been carried out in order to prove the significance of proposed approach and results shows that the turnaround time of trustworthy jobs can be reduced significantly by more than 50%.

Keywords: Grid Computing; Grid Scheduling; Trust; Job scheduling; Grid system;

I. INTRODAUCTION

Albeit speed and capacities of computing elements continue to increase, resource-intensive applications are proliferating as well. The need of large-scale computational resources for analyzing massive scientific data and distribution of knowledge are two major problems commonly observed in scientific disciplines. Even though many computers with high computational power as a whole available at many organizations, institutions and universities, the users facing shortage of processing power due to the distributive ownership of the computers. Many computers remains idle for much of the time and at the same time other users facing shortage as they require more computational power beyond the capacity of their machines. The idle machines hardware cannot be moved to these users as it is too expensive to replicate on other locations. Grids overcome these obstacles and provide the tools of research to distant users. Instead of purchasing more resources for these users, grid provides the framework to pool the unutilized resources and make them available to users who demand more resources. A grid can federate wide variety of geographically distributed computational resources such as desktops, workstations and clusters, storage systems, data sources, databases and special purpose scientific instruments and all these resources are visualized as unified integrated resource.

A. Types of Grids:

A grid can be classified into three categories: Computational grid, Data grid and Service grid. Fig. 1 shows the categories of grid systems.

A computational grid provides access to huge pool of shared processing power suitable for high throughput applications and computation intensive computing.

Depending on how the capacity is utilized, computational grid is classified into Distributed Supercomputing grid and High Throughput grid. A supercomputing grid executes applications on multiple machines in parallel to reduce the computation time of a task.

A data grid provides an infrastructure for synthesizing information from data archives such as digital libraries or data warehouses that are distributed in a wide area of network. European data grid project and Globus are data grid initiatives, working on developing large scale data organizations.

A service grid provides services that cannot be accomplished by any single machine. The service grid further categorized into on-demand grid, collaborative grid and multimedia grid systems [30].

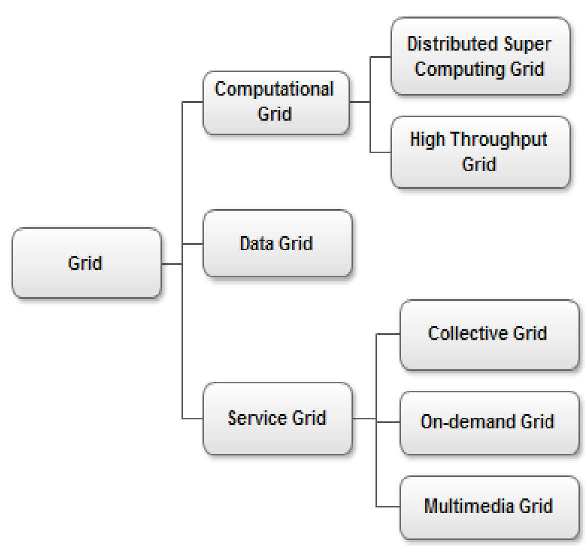

Figure 1. Grid System classification

CONFERENCE PAPER
II International Conference on
"Advance Computing and Creating Entrepreneurs (ACCE2013)"
On 19-20 Feb 2013
Organized by
2nd SIG-WNs, Div IV & Udaipur Chapter , CSI , IEEE Computer Society Chapter India Council , IEEE Student Chapter Geetanjali Institute of Technical Studies, Udaipur, Rajasthan, India

B. Grid Scheduling framework:

Even though various types of resources on the grid may be shared and used, they are usually accessed via an executing application or job. A job (or meta-task, or application) is a set of atomic tasks that will be computed on a set of resources. Jobs can have sub-jobs or tasks, and sub-jobs can themselves be decomposed further into atomic tasks. The grid system is responsible for sending a job to a computing element in the grid. More advanced grid systems include a job scheduler of some kind that automatically finds the most suitable machine on which to run any given job that is waiting to be executed. There are two kinds of schedulers in grid environment: local scheduler and grid scheduler. A scheduler or local scheduler is responsible for scheduling of jobs and managing resources at a single site (an autonomous entity composed of one or multiple resources) or node or cluster or resource. A global scheduler or grid scheduler is responsible for selecting appropriate local site and mapping of jobs onto the selected site or domain. A grid scheduling framework has been shown in Fig. 2. Due to the heterogeneous nature of grid, information about the status of available resources in the grid is very important for global scheduler to make scheduling decisions. The Grid Information Service (GIS)[26] in grid provides information about available resources such as CPU capacity, memory size, network bandwidth and load of site in a particular period to grid scheduler.

A local scheduler resides in a resource domain, it schedules not only jobs from exterior grid users but also jobs from the users of that local site and it reports the information to GIS. There can be one or more local schedulers running within a site with specified resource management policies. Some of the local schedulers are Condor [25] and Open PBS[27].

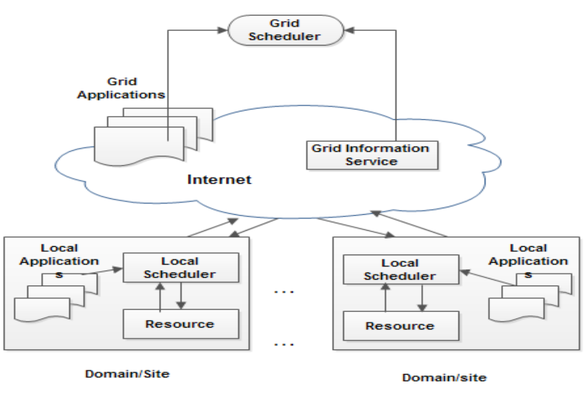

Figure 2. Grid Scheduling Framework

C. Grid Middleware:

Grid middleware is the software that provides easy access to the heterogeneous resources in the grid environment. Several Grid middleware software toolkits have been developed by many Universities as their research projects. UNICORE [17], Legion [18], Globus [19], gLite [20], CONDOR [21], LCG [22], GARUDA [23] and Gridbus [24] are some of the middleware frameworks.

The rest of this paper is structures as follows. Section II describes related work, and the section III describes about proposed approach. Empirical analysis and results have been discussed in section IV, followed by conclusion in Section V.

II. RELATED WORK

Computational grids are effective solution for executing resource intensive batch applications. But Scheduling of jobs efficiently in this computational grid platform is a complex task as grid consists of heterogeneous resources at different sites or nodes. Various algorithms have been proposed for effective scheduling of jobs at each site.

Jason et al. [3] proposed a reputation based scheduling model in which job allocation is based on the behavior of the nodes prior performance. This approach handles the unreliability of nodes in large-scale distributed infrastructure and it allocates the job to the reliable nodes in the grid. In [2] Thamaraiselvi et al. proposed a trust model to evaluate the trust of a resource based on affordability, success rate and bandwidth.

Shanyu et al. [4] also proposed a trust based scheduling model. In this model, reputation and accuracy of results a node generating is considered. It assigns a trust value to particular node based on the accuracy of the results of the jobs at that node. A job is allocated to the most trusted host based on its trust value.

A HQ-GTSM (Grid Task Scheduling Mechanism based on Heap-sort and QoS) scheduling algorithm has been proposed [8]. This algorithm not only takes into account the input jobs but also considers the resource migration time in making scheduling decisions.

Mingzhong Wang et al. [1] incorporated trust to manage life cycle of scientific workflow and to improve robustness and predictability of a schedule.

In [28] Hemamalini et al. surveyed different scheduling algorithms in grid environment. The Min-Min scheduling algorithm finds the job which has minimum execution. The Max-Min algorithm is similar to Min-Min but schedules the job which has maximum execution time first. The Minimum Completion Time scheduling algorithm schedules the job which has minimum completion time.

Kokilavani et al.[29] proposed a Load Balanced Min-Min scheduling algorithm. It first schedules the tasks based on Min-Min, then Identifies the resources with heavy load by choosing the resource with high make-span in the schedule produced by Min-Min. The completion time for the first task is calculated for all resources in the current schedule, then the maximum completion time of that task is compared with the make-span produced by Min-Min, if it is less than make-span then the task is rescheduled in the resource that produces it.

Saeed Parsa et al. proposed a Resource Aware Scheduling Algorithm (RASA) [6]. It uses the advantages of the Min-Min and Max-Min algorithms. First the numbers of available resources are calculated. If the available number of resource is even, the Max-Min grid task scheduling

CONFERENCE PAPER
II International Conference on
"Advance Computing and Creating Entrepreneurs (ACCE2013)"
On 19-20 Feb 2013
Organized by
2nd SIG-WNs, Div IV & Udaipur Chapter , CSI , IEEE Computer Society Chapter India Council ,
IEEE Student Chapter Geetanjali Institute of Technical Studies, Udaipur, Rajasthan, India

algorithm is applied to schedule the first task. Otherwise the Min-Min scheduling algorithm is applied.

Zafril Rizal et al. [7] discussed about different Priority Rule Scheduling Algorithms. The Earliest Deadline First Algorithm schedule all the jobs according to specified due date or deadline. The Earliest Release Date algorithm, assign highest priority to the job that has earliest date in the queue. The Shortest Job First (SJF) algorithm schedules the shortest job first. The Longest Job First (LJF) algorithm schedules the longest job first.

Current Grid schedulers schedule a given job according to the availability and the performance provided by resources at that moment without considering the previous knowledge of their performance. In parameter sweep applications, a job has to be executed many number of times for different types of inputs. This type of job may or may not execute within the expected execution time or expected resource requirements at every time. A job which executes within the estimated time or resource constraints is more trust worthy than other jobs and it has to be scheduled before than other jobs in the job queue at a particular site.

After having gone through all work proposed by previous researchers, we have come across following issues:

a. Trust has not been considered in any of the grid scheduling algorithm.

b. No algorithm has taken into account all the factors like trust, resource need, priority and time consumption.

c. All the previous algorithms considered the resource as trusted entity, whereas the job itself can be malicious and hold the resources for longer time.

Hence, we propose a hybrid scheduling approach that addresses above mentioned issues in the next section.

III. THE PROPOSED APPROACH

Table I shows the notations that are considered in the proposed algorithm. Jobs are scheduled at a site in grid environment based on the Scheduling Factor(SF_{Ji}). The scheduling factor of a job J_i is depends on four sub factors of a job: Trust(tr_{Ji}), Resource need(r_{Ji}), Priority(p_{Ji}) and Time consumption(t_{Ji}). All the factors are normalized to 0 to 1.

Table I. Notations and their description

Notation	Definition
J_i	Job i (i= 1,….n)
tr_{Ji}	Trust factor of job J_i
r_{Ji}	Resource need factor of job J_i
p_{Ji}	Priority factor of job J_i
t_{Ji}	Time consumption factor of job J_i
SF_{Ji}	Scheduling factor of job J_i
S_{Ji}	Number of times job J_i submitted to the grid
E_{Ji}	Number of times job J_i executed within estimated time
MA_{Rk}	Maximum availability of resource R_k (k=1,…m)
A_{Rk}	Current availability of resource R_k
J_{iRk}	Number of resources of type R_k needed by job J_i
MinETJi	Minimum execution of job J_i
MaxETJi	Maximum Execution time of job J_i

A. Assumptions:

We have considered following assumptions in our approach:

a. Enough number of resources are available to execute he jobs at local machines (nodes) in grid.
b. Only parameter sweep jobs which need to be executed different number of times are considered as an input in proposed work.
c. The global scheduler assigns the base priority to jobs.
d. In the proposed approach, the priority of processes has been considered as per the Linux systems.

B. Subfactors:

a. ***Trust Factor:*** This factor describes how much a job is trust worthy. In parameter sweep applications, a job has to be executed many times for different parameters. A job may execute within estimated time or resource requirements. A job which is executed within all estimated requirements is most trust worthy than others.

$$tr_{Ji} = \frac{E_{Ji}}{S_{Ji}} \qquad (1)$$

b. ***Resources need factor:*** This factor describes about amount of different types of resources needed for the job to complete execution. It considers maximum amount of resource available. We have considered the trust factor also here to make the resource need factor unbiased for jobs that need more resources.

$$r_{Ji} - \frac{\sum_{i=0}^{n}\left(\sum_{k=0}^{m} J_{iRk} / MA_{Rk}\right)}{M} * tr_{Ji} \qquad (2)$$

c. ***Priority factor:*** Every job in the grid system has some base priority. In Linux systems the priority ranges from 0-139.

We have considered 140 to normalize the priority factor to 0 - 1.

$$p_{Ji} = \left(1 - \frac{BasePriority}{140}\right) * tr_{Ji} \qquad (3)$$

d. ***Time consumption:*** This factor is based on the previous behaviour of the job. It considers minimum execution time and maximum execution time of the job base on its previous behaviour.

$$t_{Ji} = \frac{MinET_{Ji}}{MaxET_{Ji}} * tr_{Ji} \qquad (4)$$

Fig. 3 shows the proposed scheduling algorithm and Table 2 shows the factors considered in other grid scheduling algorithms. The proposed algorithm considers all the factors.

CONFERENCE PAPER
II International Conference on
"Advance Computing and Creating Entrepreneurs (ACCE2013)"
On 19-20 Feb 2013
Organized by
2nd SIG-WNs, Div IV & Udaipur Chapter , CSI , IEEE Computer Society Chapter India Council , IEEE Student Chapter Geetanjali Institute of Technical Studies, Udaipur, Rajasthan, India

- Input: A set of jobs ($J_1,\ldots,J_n$) and their parameters (S_{Ji}, E_i, P_{Ji}, $MinET_{Ji}$, $MaxET_{Ji}$)
- Output: Scheduling factor of jobs

Step-1: Calculation of Trust factor(tr_{Ji})
for i=1 to n
$tr_{Ji} = E_i / S_i$
Step-2: Calculation of resource need factor(r_{Ji})
for i=1 to n
t = 0
for k=1 to m
$t = t + J_{iRk} / MA_{Rk}$
$r_{Ji} = k/M * tr_{Ji}$
Step-3: Calculation of time consumption factor(t_{Ji})
for i= 1 to N
$t_{Ji} = (MinET_{Ji} / MaxET_{Ji}) * tr_i$
Step-4: Calculation of priority factor(p_{Ji})
for i=1 to N
$p_{Ji} = (1-(basePriority/140)) * tr_{Ji}$
Step-5: Calculation of scheduling factor(SF_{Ji})
for i=1 to N
$SF_{Ji} = r_{Ji} * t_{Ji} * p_{Ji}$

Figure 3. Proposed scheduling algorithm

Table 2. Factors considered in other scheduling algorithms

Factors / Scheduling Algorithm	Trust	Resource need	Priority	Time Consumption	Arrival time
FCFS					√
SJF				√	√
Round Robin				√	√
Min-Min				√	√
Priority			√		√
Max-Min				√	√
Earliest Deadline First					√
Earliest Release Date					√
Longest Job First				√	√
Proposed Algorithm	√	√	√	√	√

IV. EMPIRICAL ANALYSIS AND RESULTS

To carry out the empirical analysis, we have considered 5 jobs J_1, J_2, J_3, J_4, J_5 and 3 different types of resources R_1, R_2 and R_3. Each resource type has 10 instances and each job requires one or more instances of each resource type to complete its execution. We Table III shows the resource requirement of each job and Table IV shows history of jobs. Table V shows scheduling factors for each job that are calculated from the factors we have considered.

Based on the scheduling factor the scheduling sequence is <J_3, J_4, J_1, J_5, J_2>. After performing different simulations with different values for each behavioral parameter of the job we have observed that all the factor: trust, resource need, priority and time consumption effect the turnaround time of jobs.

The empirical analysis with different set of jobs has been conducted. Fig. 4 shows the turnaround time of jobs if untrustworthy jobs do not exist in the system and Fig. 5 shows the turnaround time of jobs if untrustworthy jobs exist in the system. It is observed that if job 3 is trust worthy and job 5 and job 2 are untrustworthy jobs then proposed approach schedules job3 first and produces less turnaround time for job3 than untrustworthy jobs. The trust worthy jobs has less turnaround time around 50% if untrustworthy jobs exist in the grid system.

Table 3. Jobs considered and their resource requirements

Job id	Requirement		
	R_1	R_2	R_3
J_1	7	5	3
J_2	3	2	2
J_3	9	0	2
J_4	2	2	2
J_5	4	3	3

Table 4. History of jobs

Job id	S_{Ji}	E_i	P_{Ji}	$MinET_{Ji}$	$MaxET_{Ji}$
J1	10	8	110	7	15
J2	8	7	115	3	7
J3	10	10	104	9	9
J4	5	4	105	2	2
J5	8	7	101	4	10

Table 5. Scheduling factors for each job

Job id	tr_{Ji}	r_{Ji}	t_{Ji}	p_{Ji}	SF_{Ji}
J1	0.8	0.5	0.3733	0.080	0.0119
J2	0.875	0.233	0.3750	0.066	0.0051
J3	1	0.366	1	0.257	0.0942
J4	0.8	0.2	0.5333	0.133	0.0113
J5	0.875	0.333	0.35	0.097	0.0099

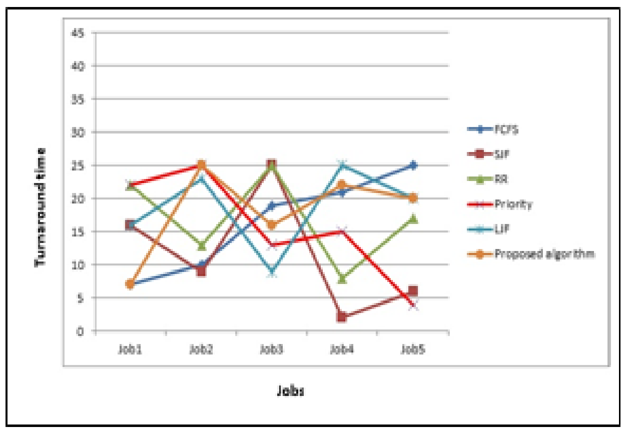

Figure 4. Turnaround time of jobs without untrustworthy jobs

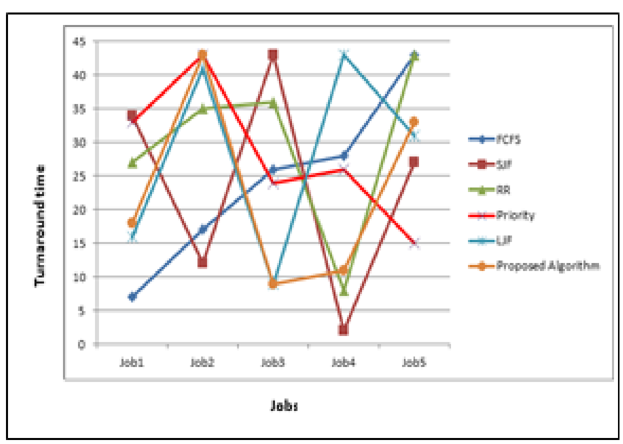

Figure 5. Turnaround time of jobs with untrustworthy jobs

V. CONCLUSION

In this paper, we have proposed trust based job scheduling algorithm in grid environment. The proposed approach considered factors like trust, resource need, priority and time consumption to schedule the jobs. An empirical study has been carried out with five jobs with different iterations. It has been observed that the proposed scheduling algorithm reduces the turnaround time of trust worthy jobs more than 50%. In future, we shall implement our proposed scheduling algorithm using GridSim tool.

VI. REFERENCES

[1]. Mingzhong Wang, Kotagiri Ramamohanarao and Jinjun Chen, "Trust-based robust scheduling and runtime adaptation of scientific workflow", Concurrency and computation: practice and experience, 2009.

[2]. S ThamaraiSelvi, P Balakrishnan, R Kumar, K Rajendar, "Trust Based Grid Scheduling Algorithm for Commercial Grids", International Conference on Computational Intelligence and Multimedia Applications, 2007.

[3]. Jason Sonnek, Abhishek Chandra and Jon Weissman, "Adaptive Reputation-Based Scheduling on Unreliable Distributed Infrastructures", IEEE Transactions on Parallel and Distributed Systems, Nov 2007.

[4]. Shanyu Zhao and Virginia Lo, "Result Verification and Trust-based Scheduling in Open Peer-to-Peer Cycle Sharing Systems", Fifth IEEE International Conference on Peer-to-Peer Computing, 2005.

[5]. M. Aggarwal, R.D. Kent and A. Ngom, "Genetic Algorithm Based Scheduler for Computational Grids", Proc. 19th Annual International Symposium on High Performance Computing Systems and Applications, pp.209- 215, May 2005.

[6]. Saeed Parsa and Reza Entezari –Maleki, "RASA: A New Task Scheduling Algorithm in Grid Environment", World Applied Sciences Journal, 2009.

[7]. Zafril Rizal M Azmi, "Performance Comparison of Priority Rule Scheduling Algorithms Using Different Inter Arrival Time Jobs in Grid Environment", International Journal of Grid and Distributed Computing, 2011

[8]. Huyn zhang, chanle wu, Q xiong, and L Wu G Ye, "Research on an Effective Mechanism of Task Scheduling in Grid Environment", Fifth International Conference on Grid and Cooperative Computing, 2006.

[9]. Deepti Malhotra and Devanand, "Framework for Job Scheduling in Grid Environment", International Journal of Computer Applications, Volume 38– No.7, Jan 2012.

[10]. K Thanushkodi, K Deeba, "A New Improved Particle Swarm Optimization Algorithm for Multiprocessor Job Scheduling", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011.

[11]. Bart Jacob, Michael Brown, Kentaro Fukui, Nihar Trivedi, "Introduction to grid computing", IBM redbooks 2005.

[12]. Fangpeng Dong and Selim G. Akl, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems", A Technical Report.

[13]. Rayburn, "7 things you should know about grid computing", A technical paper.

[14]. Jennifer M. Schopf, "Ten actions when grid scheduling".

[15]. Zafril Rizal M Azmi, Kamalrulnizam Abu Bakar, Abdul Hanan Abdullah, Mohd Shahir Shamsir, Wan Nurulsafawati and Wan Manan, "Performance comparison of priority rule scheduling algorithms using different inter arrival time jobs in grid environment", International Journal of Grid and Distributed Computing, Vol. 4, No. 3, September, 2011.

[16]. Klaus Krauter, Rajkumar Buyya and Muthucumaru Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing", Software-Practice and Experience, 2002

[17]. Tanu, "Grid Scheduling Algorithm based on Dynamic Time Quantum", Computer science and engineering department, Thapar university,2010.

[18]. Demuth B, Schuller B, Holl S, Daivandy J, Giesler A, Huber V, and Sild S, "The UNICORE rich client: facilitating the automated execution of scientific workflows",IEEE Sixth International Conference on e-Science, Dec. 2010, in press.

[19]. Lewis M and Grimshaw A, "The core legion object model", Proc. 5th IEEE International Symposium on High Performance Distributed Computing, Aug. 1996, in press.

[20]. Foster I, "The globus toolkit for grid computing", First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001, in press.

[21]. Dario Bruneo, Marco Scarpa, and Antonio Puliafito, "Performance evaluation of gLite grids through GSPNs", IEEE Transactions on Parallel and Distributed systems, volume 21, issue 11, pages 1611 - 1625, Nov 2010.

[22]. Buyya R, Venugopal S, "The gridbus toolkit for service oriented grid and utility computing: an overview and status report", 1st IEEE International Workshop on Grid Economics and Business Models, April 2004, in press.

[23]. Roberto Santinelli and Flavia Donno "Installing and configuring application software on the LHC Computing Grid", First International Conference on e-Science and Grid Computing, 2005, in press.

[24]. Arackal V S, Arunachalam B, Bijoy M B, Prahlada Rao B B, Kalasagar B, Sridharan R, Chattopadhyay S, "An accessmechanism for Grid GARUDA", IEEE International Conference on Internet Multimedia Services Architecture and Applications (IMSAA), Dec. 2009.

[25]. Condor http://research.cs.wisc.edu/condor/manual/v7.8/index.html.

[26]. Karl Czajkowski, Steven Fitzgeraldt, Ian FosterB and Carl Kesselman, "Grid Information Services for Distributed Resource Sharing", Proc. 10th IEEE International Symposium on High Performance Distributed Computing, 2001.

[27]. Sterling, T,"PBS: Portable Batch System " , Beowulf Cluster Computing with Windows, MIT Press.

[28]. M. Hemamalini, "Review on Grid Task Scheduling in Distributed Heterogeneous Environment", International Journal of Computer Applications, Volume 40– No.2, February 2012.

[29]. Kokilavani , D.I. George Amalarethinam , "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing ", International Journal of Computer Applications ,Volume 20– No.2, April 2011.

[30]. Klaus Krauter, Rajkumar Buyya and Muthucumaru Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing", Software-Practice and Experience, 2002.