Journal of Computer Science 7 (4): 493-498, 2011
ISSN 1549-3636

Performance Analysis of Resource Selection Algorithms in Grid Computing Environment

Malarvizhi Nandagopal and Rhymend Uthariaraj
Ramanujan Computing Centre, Anna University, Chennai, India

Abstract: Problem statement: Meta- scheduling has become important due to increased number of jobs and resources in the area of grid computing to achieve the objectives of grid users and grid resources. **Approach:** A variety of factors need to be considered for effective scheduling of resources in such environments such as total time job spend in the grid system, global and local allocation policies and resource utilization. A general and extensible scheduling architecture addressing these issues is proposed. Within this architecture a Multi Criteria Resource Selection (MCRS) algorithm is developed and performance of this algorithm is evaluated using simulations for a wide range of parameters. To select a resource for job execution the proposed algorithm considers multiple criteria like processing power, workload and network bandwidth of the resource. **Results:** The proposed algorithm is compared with the conventional single criteria resource selection algorithms. **Conclusion:** Simulation results show that in order to improve the performance of both user and resource it is important to consider multiple criteria together to select the optimal resource, rather than considering them separately.

Key words: Performance analysis, LARS algorithm, computing environment, Meta- scheduling, Multi Criteria Resource Selection (MCRS), Grid Information Server (GIS), Power Aware Resource Selection (PARS)

INTRODUCTION

Grid computing uses the resources of many separate computers connected by a network to solve large scale problems. The resources may involve CPU cycles, memory, bandwidth, applications and databases on remote systems. These heterogeneous resources are distributed and owned geographically by different organizations for solving large scale problems in science, engineering and commerce (Foster *et al*., 2001). The tremendously large number and the heterogeneous potential of grid resources cause the resource selection challenge to be a significant effort topic in grid environments.

Grid scheduling is defined as the process of making scheduling decisions involving allocating jobs to resources over multiple administrative domains. Grid scheduling is usually viewed as a hierarchical problem with two levels. At the first level, called meta-scheduling, a grid scheduler selects the resources to be used by a job. At the second level, called local scheduling, a local scheduler schedules the jobs assigned to it. The grid scheduler and the local scheduler differ in that the latter only manages a single resource, e.g. a single cluster with multiple machines, while the former considers more than one resource in making its decisions. In particular, the grid scheduler receives jobs from grid users and generates job-to resource assignments, by optimizing some objective function. In this study more interesting is in the first level of scheduling, the grid scheduling level. Widespread availability of low-cost, high performance computing hardware and the rapid expansion of the Internet and advances in computing networking technology have led to an increasing use of heterogeneous computing systems (Kontothanassis and Goddeau, 2005). Such systems are constructed by networking various machines with different capabilities and coordinating their use to execute jobs. The different phases of grid scheduling process have been discussed (Malarvizhi and Uthariaraj, 2008). A multi-agent architecture that addressed resource management and application execution with support for Quality of Services (QoS) in grid environment is presented (Rezaee *et al*., 2008). Grid scheduling algorithms in different occasions are discussed by (Benoit *et al*., 2005).

According to **(**Jiang *et al*., 2008), job behavior in the job waiting queue is considered as an important factor for scheduling algorithm. The data access cost is also aggregated with the job waiting queue in order to reduce the job turnaround time. A framework that includes meta-scheduling, local scheduler and dataset

Corresponding Author: Malarvizhi Nandagopal, Ramanujan Computing Centre, Anna University Chennai, India

scheduler was proposed by (Ranganathan and Foster, 2003). Regarding meta-scheduler, the data access cost and the job waiting queue are the only criteria considered in the decisions. Ernemann *et al*., (2004) proposed a meta-scheduler in which scheduler selects the resource that provides the nearest distance to the required data for the job execution to reduce the job completion time. Hence, the data access cost is the main factor which is focused in the scheduler. Ding *et al.* (2004) provide an adaptive meta-scheduler in which the best resource is the one that provides highest computation. The computational power of the resource is the main factor which is focused in the scheduler. AL-Khateeb *et al*. (2009) considered a new factor namely the job ratio which is used to reduce the job turnaround time by submitting jobs in batches rather than submitting the jobs one by one. The job execution time and the data access time for each job is monitored and computed to provide the Job-Ratio. An elaborate prediction function is produced for computing the Job-Ratio based on the history file and other grid services tools. Malarvizhi and Uthariaraj (2009) discussed about the mechanism for scheduling jobs to resources by considering multiple criteria but did not compare the performance with single criteria algorithms.

In some existing methods, the grid scheduler decides the best resource based on the number of jobs that is waiting in the queue. While other proposed methods select the resource based on the data access cost which includes data transfer time, data locations and storage access latency. Some other proposed methods select the resource based on the processing power of the resource. These resource selection methods based on any one criteria is insufficient for the best decision, because the resources are heterogeneous and administered by different administrative domains. Considering only job waiting time or file transfer time or processing power of the resource does not yield best schedule in the heterogeneous grid environment.

In this study, Multi Criteria Resource Selection (MCRS) algorithm is developed for the grid environment. The overall objective of the scheduler presented here is to select the resource (cluster) that gives the shortest time for job completion for each job, including the time for file staging (file transferring), resource queue waiting and job execution. The performance of the proposed MCRS algorithm is compared with the following single criteria resource selection algorithms:

- Load Aware Resource Selection (LARS): Job is scheduled to the resource that has the least workload. Load is simply defined as the number of jobs waiting to run
- Bandwidth Aware Resource Selection (BARS): Job is scheduled to the resource that has more bandwidth
- Power Aware Resource Selection (PARS): Job is scheduled to the resource that has more computational power.

MATERIALS AND METHODS

In topological point of view, grid is considered as a collection of heterogeneous clusters as shown in Fig. 1. There is a global scheduler (Grid Scheduler) which communicates with each one of the distributed clusters. For simplicity it is assumed that each cluster has two nodes and each node has one machine and each machine has one processing element. The scheduler serves jobs in the FCFS order.

Some of the important components in the scheduling framework are described below:

GIS contains information about all available grid resources. Whenever a scheduler has jobs to execute, it consults GIS to get information about available grid resources. Grid users register themselves to the GIS of a grid by specifying QoS requirements. The QoS requirement consists of deadline (D), number of nodes required and size of each job measured in Millions Instructions (MI).

Grid scheduler: Grid scheduler is responsible for receiving jobs from grid users, selects feasible resources for those jobs according to acquired information and finally generates jobs - to- resource mappings. The entities of scheduler are Resource Selector, Matchmaker, TTR Estimator and Job Dispatcher. The matchmaker entity performs match making of the resources and job requirements by contacting GIS and filter out the resources that satisfies job requirements. TTR estimator estimates the total time spent by the job on each matched resource based on transfer time, queue wait time and execution time of the job. Resource selector selects the resource with minimum TTR. Then the job dispatcher dispatches the job one by one to the selected resource.

MCRS algorithm:

- Generate a list of all individual requests by validating the user specification(s)
- Contact GIS to obtain a list of available resources
- Query each resource to obtain static and dynamic information such as hardware and software characteristics, current status, work load and so on
- For each job J_i with deadline D_i from a queue do

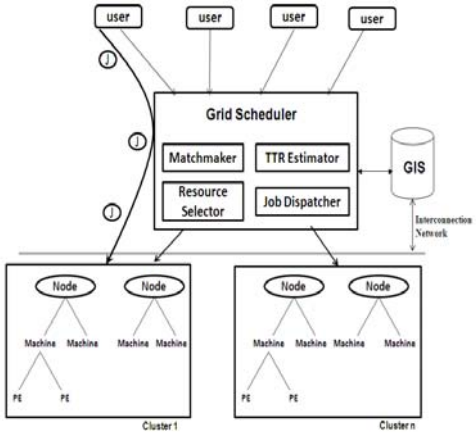

Fig: 1. Grid Scheduling Framework

- Filter out the resources that do not fulfill the job requirements and also the unauthorized users
- For each filtered resource R_j at which this job can be executed do
- Retrieve estimated TTR of job $TTR_{Ji, Rj}$
- Filter out the resource R_i in which $TTR_{Ji, Ri} > D_i$
- Assign job Ji to resource R_{best} so that $TTR_{Ji, Rbest} = \underset{R_j}{Min} (TTR_{Ji,,Rj})$
- end for
- end for

Estimation of TTR: The TTR for a job j in resource r includes transmission time of input and output data to and from r, waiting time of j in r and the service time of j when assigned to r. For simplicity TTR is expressed as t_{total} and is defined by Eq. (1):

$$t_{total}^{jr} = t_{transfer}^{jr} + t_{wait}^{jr} + t_{process}^{jr} \quad (1)$$

The estimated transmission time of the j from the scheduler to the resource r can be determined by Eq. (2):

$$t_{transfer}^{jr} = \frac{S_j}{BW_r} \quad (2)$$

where S_j is the size of the job j and BW_r is the network bandwidth between the scheduler and the resource r.

Processing Time of the job j in resource r is defined by Eq.(3):

$$t_{process}^{jr} = \frac{S_j}{PP_r} \quad (3)$$

where PP_r represents the processing power of the resource r.Waiting Time of the job j in the resource queue r could be estimated by the sum of all processing time of jobs in waiting queue of r which has been assigned to that resource before arriving of job j and the remaining processing time of job j-1 executed in the resource and is defined by Eq. (4):

$$t_{wait}^{jr} = \sum_{i=1}^{n} t_{process_r}^{i} + t_{remaining_r}^{j-1} \quad (4)$$

where n is the number of jobs in the resource queue. The remaining time is calculated by Eq.(5):

$$t_{remaining_r}^{j-1} = t_{process_r}^{j-1} - (t_{sub_r}^{j-1} - t_{cur_r}) \quad (5)$$

where $t_{sub_r}^{j-1}$ represents submission time of job j-1 in resource r and t_{cur_r} represents current time of resource r. MCRS algorithm selects the resource r for job j which gives minimum TTR and is expressed by Eq.(6):

$$R_{select} = Min \{ t_{total}^{jr} \} 1<j<n, 1<r<m \quad (6)$$

Where n represents number of jobs and m represents number of resources.

RESULTS AND DISCUSSION

The proposed MCRS algorithm is compared with single criteria resource selection algorithms under different system parameters. Simulation parameters are listed in Table 1. The setup makes this grid environment non dedicated, heterogeneous and dynamic.

Effect of grid size: The following experiments are carried out to test the scalability effect of the grid size. Figure 2-4 compares the performances of LARS, BARS, PARS and MCRS under different grid sizes. In this experiment grid sizes vary from 10 to 50 clusters (S = 10, 20, 30, 40, 50).

Initially jobs are processed in the grid consisting of the smallest size (S = 10, 20), then more clusters are added to the grid environment until all 50 clusters are used. In this experiment how different grid size takes the effect on TTR is considered. To check the performance, 1000 jobs are submitted. From the results in Fig. 2, for all the four strategies, smaller grid size leads to higher TTR. But when the size of the grid is larger, TTR decreases quickly for all the algorithms. Under a small size grid (S = 10) the TTR using MCRS and LARS are close to each other but MCRS can be as much as 78%, 75% shorter than that using PARS, BARS respectively. When the size of the grid increases(S = 50) the TTR of MCRS can be as much as 17% shorter than LARS. Figure 3 shows the throughput

with varying number of grid resources. MCRS achieves good throughput in a small size grid scenario with LARS, PARS and BARS closely behind. When the grid size reaches 50, throughput of MCRS increases quickly as job requirements can be satisfied by many resources. It can be observed from Fig. 4 that MCRS outperforms LARS, PARS and BARS in terms of resource utilization in all cases. A further observation is that the advantage of MCRS over LARS, BARS and PARS are more pronounced as the grid size increases. Table 2 shows the performance improvement using MCRS over PARS, BARS and LARS when grid size is 50.

Table 1: Simulation parameters

Parameter	Value
Number of clusters	10-50
Number of nodes/cluster	4-12
No. of processor/node	1
No. of nodes required	4-8
Processing power/node	277-577 MIPS
Job Length	1000000-100, 00, 000 MI
Input and Output file size	5000-7000
No. of Queue/Cluster	1
Number of jobs	100-800
Deadline	10 -100 Seconds
Bandwidth	100KB-1MB

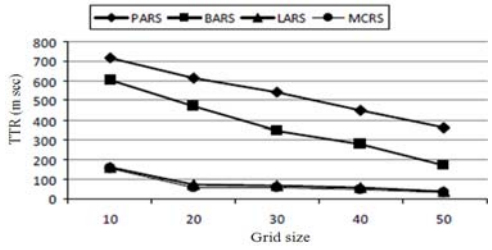

Fig. 2: TTR with Grid Size

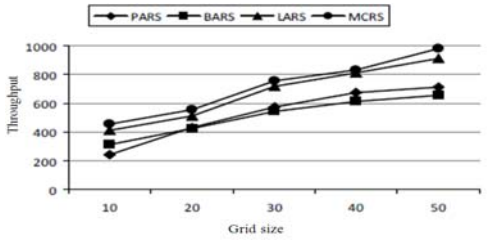

Fig. 3: Throughput with grid size

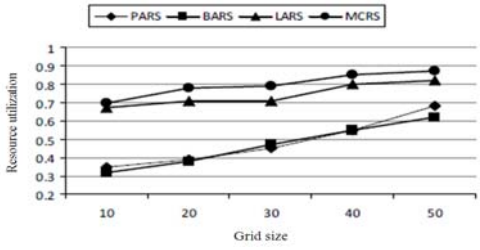

Fig. 4: Resource utilization with grid size

Effect of job-to-resource ratio: Job-to-resource ratio, denoted as Rjr is the ratio of the number of jobs and the number of resources. When the numbers of jobs and resources are equal, the job-to-resource ratio equals 1(R_{jr} =1). When there are more jobs than resources, the job-to-resource ratio is more than 1(R_{jr}>1). When there are more resources than jobs, the job-to-resource ratio is less than 1(R_{jr}<1).

As shown in Fig. 5, the TTR goes down as the job-to-resource ratio is decreased. The reason is that the more the grid resources offered, the lesser that the jobs need to wait for getting the resource. PARS, BARS, LARS and MCRS present good results when the value of job-to-resource ratio is small. But when the value of the ratio is larger, TTR using PARS and BARS increases quickly; the TTR using of MCRS can be 13% shorter than that using LARS algorithm. When the number of jobs increases, many jobs will have to wait for a long time to find a match in PARS, BARS and LARS. The MCRS outperforms the BARS, PARS and LARS algorithms. From the result of Fig. 6, before R_{jr}>2, the throughput will increase as the job-to resource ratio is increased. After R_{jr}>2, the number of jobs is higher than the capacity of the resource, the throughput might even decrease since time is wasted on request that cannot be processed and are eventually discarded. The throughput of BARS and PARS decreases quickly; the throughput of MCRS can be 17% more than that using LARS algorithm. Considering the resource utilization, from the results in Fig. 7, before R_{jr} >2, the resource utilization increases, as job-to-resource ratio increases. After R_{jr} >2, as job-to-resource ratio is higher, the resource utilization becomes lower. Because when job-to-resource ratio increases quickly, the number of grid resources is not enough to be allocated to the jobs. Table 3 shows the performance improvement using MCRS over PARS, BARS and LARS when R_{jr} is 10.

Table 2: Improvement using MCRS when S = 50

Parameter	Algorithms PARS (%)	BARS (%)	LARS (%)
TTR	91	83	17
T	27	33	7
RU	19	25	5

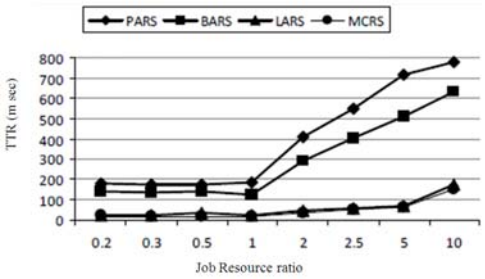

Fig. 5: TTR with job resource ratio

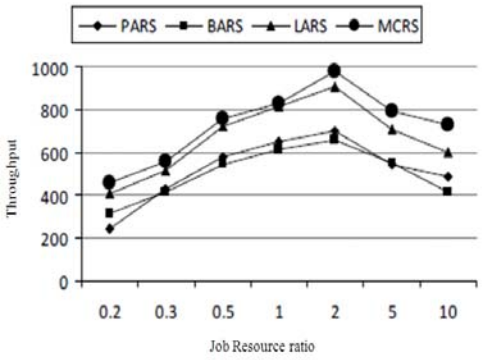

Fig. 6: Throughput with job resource ratio

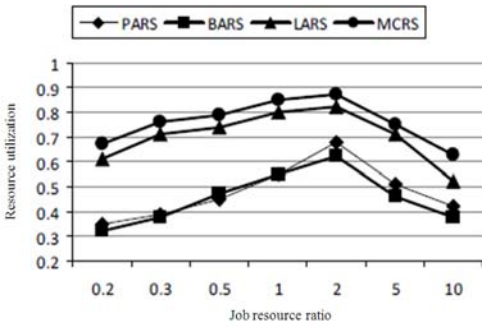

Fig. 7: Resource utilization with job resource ratio

Table 3: Improvement using MCRS when Rjr =10

Parameter	Algorithms PARS (%)	BARS (%)	LARS (%)
TTR	80	76	13
T	33	43	17
RU	33	39	17

Effect of deadline: How the user deadline affects TTR, throughput and resource utilization is illustrated in Fig. 8-10 respectively. This experiment is simulated with 1000 jobs, 40 resources having different resource configurations and the user deadline (d) is varied from 20-70 sec.

Figure 8 reveals the impact of different deadline constraints on the TTR. When the deadlines are small, all the algorithms take more time to complete jobs, because the jobs can get few resources. Larger deadline enable jobs to get more resources and obtain the result in less time. When the deadline increases MCRS outperforms the LARS, BARS and PARS algorithms because MCRS considers multiple criteria to select the resource, so the incoming jobs can be scheduled to multiple resources than other algorithms. From Fig. 9 it can be inferred that when increasing

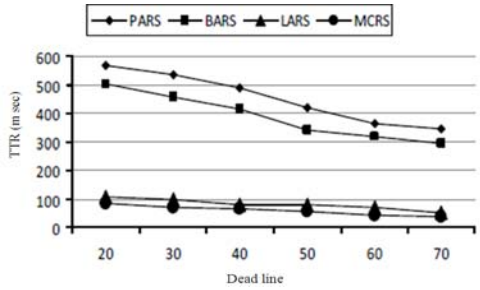

Fig. 8: TTR with deadline

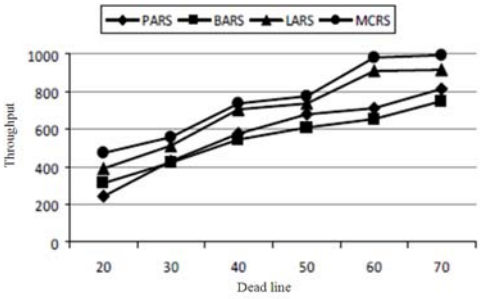

Fig. 9: Throughput with deadline

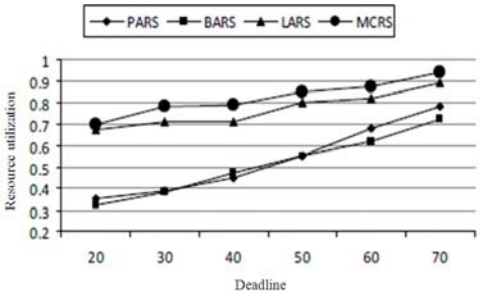

Fig. 10: Resource utilization with deadline

Table 4: Improvement using MCRS when d=70

Parameter	Algorithms PARS (%)	BARS (%)	LARS (%)
TTR	89	87	24
T	18	24	7
RU	16	22	5

deadline the throughput becomes higher. Increasing the deadline will facilitate the number of jobs to be admitted by the system. When increasing the deadline by d = 50, the throughput in MCRS is 39% more than that with d = 20.When d = 70, the throughput increases to nearly 52% compared with d=20. Considering the resource utilization, from the results in Fig 10, as the deadline is higher, the resource utilization becomes higher. When d =70, the resource utilization in MCRS

is 24% more than utilization by d=20, because when the deadline decreases quickly the jobs will be prevented from obtaining resources. Table 4 shows the performance improvement using MCRS over PARS, BARS and LARS when d is 70.

CONCLUSION

In this study the performance of proposed MCRS algorithm is compared with the conventional single criteria resource selection algorithms. The experimental results demonstrate that proposed algorithm effectively schedule the grid jobs by considering multiple criteria in spite of highly dynamic nature of grid. The analysis clearly reveals that it is important to consider multiple criteria together to select the optimal resource, rather than considering them separately. Throughout all the varied experimental scenarios simulated, the proposed MCRS algorithm maintains its superiority over many single criteria resource selection algorithms.

REFERENCES

AL-Khateeb, A., R. Abdullah and N.A. Rashid, 2009. Job type approach for deciding job scheduling in grid computing systems. J. Comput. Sci., 5: 745-750. DOI: 10.3844/jcssp.2009.745.750

Benoit, A., M. Cole, S. Gilmore and J. Hillston, 2005. Enhancing the effective utilisation of grid clusters by exploiting on-line performability analysis. Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, May 9-12, Edinburgh University, UK., pp: 317-324. DOI: 10.1109/CCGRID.2005.1558571

Ding, S.L., J.B. Yuan and J.B. Ju, 2004. An algorithm for agent-based task scheduling in grid environments. Proceedings of the International Conference on Machine Learning and Cybernetics, Aug. 26-29, Jilin University, Changchun, China, pp: 2809-2814. DOI: 10.1109/ICMLC.2004.1378510

Ernemann, C., V. Hamscher and R. Yahyapour, 2004. Benefits of global grid computing for job scheduling. Proceedings of the 5th IEEE/ACM International workshop on Grid Computing, Nov. 8-8, IEEE Computer Society, USA., pp: 374-379. DOI: 10.1109/GRID.2004.13

Foster, I., 2001. The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Perfomance Comput. Appl., 15: 200-222. DOI: 10.1177/109434200101500302

Jiang, J., H. Ji, G. Xu and X. Wei, 2008 .Scheduling algorithm with potential behaviors. J. Comput., 3: 51-59. DOI: 10.4304/jcp.3.12.51-59

Kontothanassis, L. and D. Goddeau, 2005. Profile driven scheduling for a heterogeneous server cluster. Proceeding of the IEEE International Conference Workshops on Parallel Processing, June 14-17, IEEE Xplore, pp: 336-345. DOI: 0.1109/ICPPW.2005.73

Malarvizhi, N and V.R. Uthariaraj, 2008. A broker-based approach to resource discovery and selection in grid environments. Proceedings of the IEEE International Conference on Computer and Electrical Engineering, Dec. 20-22, IEEE Xplore, Phuket, pp: 322-326. DOI: 10.1109/ICCEE.2008.149

Malarvizhi, N. and V.R. Uthariaraj, 2009. A new mechanism for job scheduling in computational grid network environments. Active Media Techonol., 5820: 490-500. DOI: 10.1007/978-3-642-04875-3_50

Ranganathan, K. and I. Foster, 2003. Simulation studies of computation and data scheduling algorithms for data grids. J. Grid Comput., 1: 53-62. DOI: 10.1023/A:1024035627870

Rezaee, A., A.M. Rahmani, S. Parsa and S. Adabi, 2008. A multi-agent architecture for qos support in grid environment. J. Comput. Sci., 4: 225-231. DOI: 10.3844/jcssp.2008.225.231