

Performance Evaluation of Weighted Round Robin Grid Scheduling

N.Krishnamoorthy
Assistant Professor
(Senior Grade)
Department of CSE
Kongu Engineering College
Perundurai, Erode-638052
Tamilnadu

R.Asokan, PhD.
Principal
Kongunadu College of
Engineering and Technology,
Tholurpatti, Thottiam,
Thiruchirapalli - 621215
Tamilnadu

S.Sangeetha
Assistant Professor
Department of Computer
Technology-PG
Kongu Engineering College
Perundurai, Erode-638052
Tamilnadu

ABSTRACT

Grid systems interconnect heterogeneous and geographically distributed resources to form a network which satisfy user's needs. Resource Management is its central component and involves managing system resources. It is responsible for accepting user's requests and matching it to available resources which can be accessed by the user. Schedulers are applications which manage jobs including allocating resources for specific jobs. When there are many processes in a queue, the order in which jobs are executed is decided by a scheduling algorithm. This paper proposes and investigates the performance of varied task execution for proposed weighted round robin scheduling algorithm. Simulations evaluate the proposed method's performance and results demonstrate that the proposed method performs satisfactorily.

Keywords: Grid computing, Schedulers, Resource management, Round Robin scheduling.

1. INTRODUCTION

"Grid computing" ensures large-scale resource sharing, wide-area communication and multi-institutional collaboration. "Open Grid Services Architecture" helps service and resources integration across distributed heterogeneous dynamic virtual organizations helping users with an easy request grid services [1] platform. The resource management system (RMS) controls service/resource integration, request and management. Grid computing is harnessing computer systems to solve problems cooperatively, requiring huge data storage and processing which is more than what a single system can handle [2]. Initially used in technical and scientific projects, grid computing is now popular in the corporate world due to its cost-savings potential.

In a grid system, an end user submits to the management system the job to be executed along with some constraints like job execution deadline, the maximum cost of execution. The function of the resource management is to take the job specification and from it estimate the resource requirements like the number of processors required, the execution time, and memory required. After estimating the resource requirements RMS discovers available resources and selects appropriate resources for job execution. It also finally schedules jobs on these resources by interacting with the

local resource management system.

A RMS also names system resources, monitors and reports jobs, resource status and accounts for resource usage. RMS enables a security system to validate user requests, the information service to know about resource availability, and schedules jobs through the local resource management system. Three phases of grid scheduling [3] include Resource discovery, Scheduling and job completion. Resource discovery starts a list of potential resources, while resource Scheduling pair's jobs with application requirements [4] and Job Completion includes file staging and executing clean-up.

Grid resource management face many issues because of the nature of Grid environment that complicates scheduling tasks [5] some of which include:

Resource discovery process is required as resources are geographically distributed.

- Grid resources are dynamic as resources move in and out of the grid. Thus, resource information should be regularly updated to reflect status change.

- Grid resources are heterogeneous with different architectures and operating systems and hence resource management should allocate jobs to suitable resources.

- Each resource's local access and security policies vary and this should be supported by the management model.

- Grid security is important to encourage resource providers and users in grid participation without fearing any attack. Authentication/authorization should safeguard resources and jobs against attacks.

- Users submitting jobs for execution have no control over their jobs resulting in inaccurate completion time prediction leading to some not meeting their deadlines.

The following are various grid scheduling schemes which schedule jobs on resources. Centralized scheduling scheme: In this, a central processor (centralized domain) takes charge of scheduling; the scheduler has information on all domains and their resources [6]. All domain jobs are submitted to the centralized scheduler that regulates jobs to suitable resources. Centralized scheduling ensures simple structure and easy maintenance. Grid domains are uninvolved in scheduling and send information to the scheduler about resources like availability, speed and memory. The former decides job flow based on information forwarded by domains. Centralized scheduling's advantages are it un-scalability due to voluminous information retained by centralized scheduler,

and if scheduler fails, user-resource provider interaction stops as there is only one scheduler. Jobs also suffer from long access delay as jobs are submitted to the same scheduler.

Hierarchical scheduling scheme: This is based on job scheduling's layered structure. [7]. Here jobs submitted to a central scheduler are forwarded to domains which meet their requirements. After forwarding jobs, the central scheduler has no control on such jobs. Different scheduler levels execute domain defined scheduling policies. This scheme's advantage is that each domain can use its own scheduling policy which can differ from other scheduling domains. Inability to reschedule jobs on locating a better resource is its only disadvantage.

Distributed scheduling scheme: Distributed scheduling has no central or hierarchical schedulers [6]. Each domain has its own scheduler, and interacts with domains periodically or when a job is executed. When a job needs to be executed regularly, the local scheduler assigns job to suitable local resource or to another domain resource if it is more suitable than local resources. Centralized scheduler issues like scalability, reliability, easy implementation and no single point of failure are addressed by the distributed scheduler. Literature proposes many distributed scheduling algorithms which are adaptable to resource usage changes [8].

Schedulers are applications responsible for job management including allocating resources for a specific job, partitioning jobs to ensure parallel task execution, data management, event correlation, and service-level management capabilities [9]. Schedulers are structured hierarchically with meta-schedulers forming the root and lower level schedulers and simultaneously providing specific scheduling capabilities that become leaves. Schedulers can be built with a local scheduler implementation approach for execution of particular jobs or another meta-scheduler or a cluster scheduler for parallel executions.

Jobs with Grid Computing schedulers are evaluated on a service-level requirement basis, and reallocated to respective resources for execution involving complex workflow management and data movement activities regularly. Schedulers should provide capabilities for areas like:

- Advanced resource reservation
- Service-level agreement, validation and enforcement
- Job and resource policy management/enforcement for best turnaround times within budget constraints
- Monitoring job execution/status
- Rescheduling and corrective action for partial failover situations

Though Grid computing has advanced much, QoS is an issue as Grid systems cannot be scaled proportionately to user expectations. A computational grid works in a dynamic environment with resources like bandwidth and processor time availability changing continuously to ensure no guaranteed QoS. Global applications grid applications also compete for shared resources leading to QoS degradation [10].

Grid service performance is directly linked to the collective workload on many processors globally and on participating grid sites. Predicting workload completion time is very challenging task [11, 12]. Scaling many processors to complete collective work load is not an option as bandwidth has a crucial role for data intensive workloads involving heavy data transfer loads.

When a large number of jobs are presented for Grids, these applications take overall processing including high overhead time and cost in terms of: to and from Grid resources, the job transmission and at the Grid resources, the job processing. In this paper, it is proposed to investigate the performance of schedulers for executing different number of tasks; round robin and weighted round robin scheduling algorithm are evaluated. The rest of the paper is organized as follows: Section II reviews some of the related works in literature, Section III describes the experimental setup and section IV discusses the results and Section V concludes this paper.

2. RELATED WORKS

Muthuvelu et al., [13] introduced a scheduling strategy to perform dynamic job grouping activity during runtime. Job processing granularity size is presented to allow job grouping activity to identify overall jobs to be processed at a resource at a specific time. Job grouping aims to minimize the total processing time and cost. The small scaled user jobs are grouped as few job groups considering available grid resources processing capabilities by the proposed strategy that lowers communication overhead and processing overhead times of every user job.

William M. Jonesy et al., [14] introduced a bandwidth-centric job communication model which captures interaction when applied across multiple clusters and effects co-allocating jobs simultaneously. This model is compared with earlier models which opt for a fixed execution time penalty for such co-allocated jobs. This paper also presents many bandwidth-aware co-allocating meta-schedulers. Performance of multi-cluster scheduling algorithms is estimated with a bandwidth-centric parallel job communication model harnessing time-varying utilization of shared inter-cluster network resources. Different algorithm were employed with co-allocating jobs when allocating a large fraction (85%) on a single cluster provides top performance in lowering effects that co-allocated jobs meet with due to inter-cluster network saturation slow down.

Different systems allocate resources using market mechanisms, but performance was not studied properly. Gomoluch, et al., [15] examined scenarios which outperform a traditional round-robin technique obtained through market-based resource allocation with continuous double auctions and proportional share procedure, equally. A model is developed for servers, clients and the market and simulation results discussed. The results, limited to independent tasks allocation are: 1) Continuous Double Auction Protocol (CDA) performs best in a cluster of homogeneous resources and slight communication delays, 2) Proportional Share Protocol (PSP) has similar performance to CDA during low load and has less difference among the three protocols, 3) Round-Robin performs worse than both market-based protocols with heterogeneous resources. PSP works better during high communication delays. Thus in most cases, CDA is the best protocol.

Volker Hamscher et al., [16] investigated typical scheduling structures in computational grids. Scheduling algorithms and selection strategies are presented and classified by these structures. To estimate these features on combining various Job and Machine Models, discrete-event simulation was performed. Results are discussed quantitatively and qualitatively. Backfill's importance for hierarchical scheduling is proven from through simulation. Use of a central job-pool led to unexpected results as FCFS confirmed better performance than Backfill.

Bansal et al., [17] proposed a novel grid-scheduling heuristic that schedules tasks adaptively and dynamically without requiring prior workload information on incoming tasks. The approach models the grid system as a state-transition diagram, using a prioritized round-robin algorithm with task replication to schedule tasks optimally, using prediction information on individual nodes processor utilization. Simulations, comparing the proposed approach with the round robin heuristic revealed the heuristic to be better at scheduling tasks when compared to the latter.

Hiranwal et al., [18] proposed "Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice", a Priority Driven Scheduling algorithm based on burst processes time. Processes are arranged based on execution time/burst time in an ascending order; the smaller the burst time, higher the priority of running process. This approach's idea is to select a smart time slice (STS) depending on many processes. The smart time slice equals mid process burst time of all CPU burst time when having odd number of processes. When this number is even then the time quantum is chosen according to average CPU burst of total running processes. Based on experiments/calculations the proposed algorithm radically solved fixed time quantum problem usually considered a challenge for Round Robin Scheduling Algorithm. The use of scheduling algorithm increased operating system performance and stability and built support for a self-adaptation operating system, meaning that the system will adapt itself to user requirements and not vice versa.

3. METHODOLOGY

When many jobs are presented to Grids, they take overall processing including high overhead time and cost in terms of: to and from Grid resources, job transmission at Grid resources and job processing. CPU scheduler executes processes when they have schedules. The algorithm which decides order of execution when there are many processes in a ready queue is the scheduling algorithm. Various well known CPU scheduling algorithms are First Come First Serve (FCFS), Shortest Job First (SJF) and Priority scheduling [19] all of which are non-pre emptive and unsuitable for time sharing systems. Shortest Remaining Time First (SRTF) and Round Robin (RR) are pre-emptive in nature with RR being highly suitable for time sharing systems.

Round Robin (RR) algorithm overcomes this by assigning time intervals called quantum to jobs when they are run. If a job is incomplete during a quantum it reverts back to the queue awaiting the next round [20]. The only challenge with this algorithm is finding a

suitable quantum length. Round Robin Algorithm drawbacks are that it gives equal time to all processes (processes are scheduled in a first come first serve manner) as Round Robin Algorithm drawbacks ensure it is inefficient for processes with smaller CPU bursts leading to increased waiting and response times thereby lowering system throughput. The proposed algorithm eliminates drawbacks of round robin algorithm implementation by scheduling processes through weight assignment. The proposed Weighted Round Robin algorithm depends on:

1. Number of hops from task allocating server to job performing cluster.

2. Average bandwidth between allocation server and cluster

Weighted round robin algorithm's performance is compared to simple RR for specific resource cluster number and varying tasks number.

4. EXPERIMENTAL SETUP AND RESULTS

Simulations were carried out in Simgrid framework. The simulation parameters are shown in table 1.

Table 1: Simulation parameters

| | |
|---------------------------------------|---|
| Number of node clusters | 5 |
| Number of jobs used in the simulation | 100,200,300,400,500 jobs of uniform size. |
| Job workload | Uniform size |
| Job failure probability- ρ | 0.2 |
| Scheduling schemes used | Round Robin and Weighted Round Robin. |

The simulations were conducted using 5 clusters of The resources are located at different locations connected using switches. The resources are scheduled using Round robin scheduling algorithm and the proposed weighted round robin scheduling algorithm. The number of jobs of uniform size is varied from 10 to 500. The Simgrid test bed master process created to assign tasks to available resources in round robin mode. The steps involved in round robin mode are explained as follows.

Preliminary

- (i) Initiate message_launch_application
- (ii) Calculate number of tasks to distribute
- (iii) Compute size of each task
- (iv) Compute size of files associated with the task
- (v) Identify resource (Cluster) available for assigning task
- (vi) Initialize variables to zero for Number of task, communication size and computation size

- (vii) Task creation
- (viii) Assign computation size, communication size for the task
- (ix) Process organization
- (x) Identify Resource available
- (xi) Assign task to resource
- (xii) Intimate assigned task to Server
- (xiii) As resource finishes a task, it is ready to accept next task
- (xiv) Once all task are completed, inform resources that computation is complete

Return

Table 2 tabulates part of the simulation results of the time taken to execute the varying number of tasks for Round Robin (RR) and Weighted Round Robin (WRR). Figure 1 shows the same.

TABLE 2. SIMULATION RESULTS FOR TIME TAKEN TO EXECUTE VARYING NUMBER OF TASKS

| Number of tasks | Round Robin (RR) | Weighted Round Robin (WRR) |
|-----------------|------------------|----------------------------|
| 10 | 2.41724 | 2.41724 |
| 50 | 19.7992 | 17.4503 |
| 100 | 41.1152 | 37.5915 |
| 150 | 64.7802 | 54.6829 |
| 200 | 87.0025 | 74.8248 |
| 250 | 106.709 | 91.5223 |
| 300 | 128.025 | 112.058 |
| 350 | 151.69 | 129.149 |
| 400 | 173.912 | 148.9 |
| 500 | 214.935 | 185.35 |

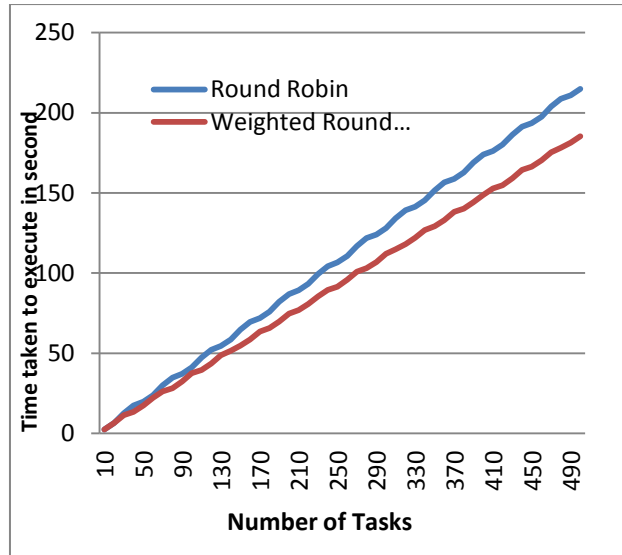


FIG.1 Time Taken to execute varying Number of Tasks

It is observed from Figure 1, the time required to carry out the scheduled tasks is comparatively lower for the proposed weighted round robin. The round robin technique produces a near linear output compared to the weighted round robin technique where nodes with higher processing power and lower number of hops are given higher preference to nodes that are can be reached only with multiple hops with constrained bandwidth. The assigning of weights with respect to number of hops and bandwidth during scheduling streamlines the process leading to efficient scheduling in less time.

5. CONCLUSION

Scheduling's goals are achievement of high performance computing and high throughput computing. The former is achieved through every jobs execution time reduction and generally used for parallel processing. This paper proposes to investigate time to execute various tasks for a novel weighted round robin scheduling algorithm. The proposed weighted round robin scheduling algorithm considers the hops number and bandwidth between server and the resource clusters. Simulation results prove that proposed scheduling improves grid performance. The performance improvement of the proposed technique improves over the round robin scheduling by 15.96%.

6. REFERENCES

- [1] Grimshaw, Andrew, Mark Morgan, Duane Merrill, HiroKishimoto, Andreas Savva, David Snelling, Chris Smith, and Dave Berry: An open grid services architecture primer, Computer 42, no. 2, 27-34, 2009.
- [2] Yu, J., &Buyya, R.: A taxonomy of workflow management systems for grid computing., Journal of Grid Computing, 3(3), 171-200,2005.
- [3] Jennifer M. Schopf: A General Architecture for Scheduling on the Grid, special issue of JPDC on Grid Computing, April, 2002.
- [4] David Fernández-Baca. "Allocating modules to processors in a distributed system", IEEE Transactions on Software Engineering, 15(11):1427-1436, 1989, November.

- [5] Zhu, Y. : A survey on grid scheduling systems, Department of Computer Science, Hong Kong University of science and Technology, 2003.
- [6] Xhafa, F., & Abraham, A.: Computational models and heuristic methods for Grid scheduling problems, *Future Generation Computer Systems*, 26(4), 608-621.
- [7] Kurowski, K., Oleksiak, A., Piątek, W., & Węglarz, J.: Hierarchical scheduling strategies for parallel tasks and advance reservations in grids. *Journal of Scheduling*, 1-20, 2011.
- [8] Mohsenian-Rad, A., Wong, V. W., Jatskevich, J., Schober, R., & Leon-Garcia, A.: Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *Smart Grid*, *IEEE Transactions on*, 1(3), 320-331, 2011.
- [9] Batista, D., & da Fonseca, N.: A survey of self-adaptive grids. *Communications Magazine*, *IEEE*, 48(7), 94-100, 2010.
- [10] Lizhe Wang, Gregor von Laszewski and Marcel Kunze : Grid Virtualization Engine: Design, Implementation, and Evaluation" *IEEE SYSTEMS JOURNAL*, VOL. 3, NO. 4, 2009, December.
- [11] AuverGrid Workload Report, http://gwa.ewi.tudelft.nl/pmwiki/reports/gwa-t-4/trace_analysis_report.html, 2009.
- [12] J. Schopf and F. Berman, "Performance Prediction in Production Environments," *Proc. 12th Int'l Parallel Processing Symp.*, pp. 647-653, 1998, April.
- [13] N. Muthuvelu, J. Liu, N. L. Soe, S. rVenugopal, A. Sulistio and R. Buyya, A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids, *Proceedings of the 3rd Australasian Workshop on Grid Computing and e-Research (AusGrid 2005)*, Newcastle, Australia, 2005, January 30 – February 4.
- [14] W. M. Jones, W. B. L. III, L. W. Pang, and D. C. S. Jr. Characterization of bandwidth-aware meta-schedulers for coallocating jobs across multiple clusters. *The Journal of Supercomputing*, 34(2):135–163, 2005.
- [15] J. Gomoluch and M. Schroeder : Performance evaluation of market-based resource allocation for grid computing. *Concurrency and Computation: Practice and Experience*, 16(5):469–475, 2004.
- [16] Hamscher, V., Schwiegelshohn, U., Streit, A. and Yahyapour, R., : Evaluation of Job-Scheduling Strategies for Grid Computing. in *7th International Conference of High Performance Computing*, (Bangalore, India), 2010.
- [17] Bansal, S., Kothari, B., & Hota, C. : Dynamic Task-Scheduling in Grid Computing using Prioritized Round Robin Algorithm. *International Journal of Computer Science*, 2010.
- [18] Hiranwal, S., & Roy, K. C. : Adaptive Round Robin scheduling using shortest burst approach, based on smart time slice. *International Journal of Computer Science and Communication*, 2(2), 319-323, 2011.
- [19] Pinedo, M. L.: *Scheduling: theory, algorithms, and systems*. Springer, 2012.
- [20] Y.-H. Lee, S. Leu, and R.-S. Chang :Improving job scheduling algorithms in a grid environment. *Future Generation Computer Systems*, 27(8):991, 2011, October.