AENSI Journals
Australian Journal of Basic and Applied Sciences
ISSN:1991-8178
Journal home page: www.ajbasweb.com

LB Scheduling for Advanced Reservation and Queuing Using TBRA in Grid Computing Environments

[1]Venkatesan. R and [2]Dr. Thanushkodi. K

[1]Assistant professor, Karunya University, Department of Information Technology, Venkatesan.R , Box.3030. Coimbatore. India.
[2]Director, Akshaya College of Engineering and Technology, Dr.Thanushkodi.K, Box.3030. Coimbatore, India.

ARTICLE INFO
Article history:
Received 12 January 2014
Received in revised form 20 March 2014
Accepted 25 March 2014
Available online 2 April 2014

Keywords:
Load Balancing, Load and time prediction, Machine learning, Scheduling, Task management.

ABSTRACT

Background: LB Scheduling for Advanced Reservation and Queuing Using TBRA in Grid Computing Environments. **Objective:** The objective of the paper is to correlate diverse load balancing techniques by determining strengths and weakness in the considered approaches and thereby propose a novel approach to balance workloads in grids. It presents a correlative investigation of the performance and efficiency of various parallel concepts that are common to one or more of the considered approaches. The proposed architecture exploits a pattern based technique to determine the type of load which is crucial in bringing about load balance and also incorporates a trust agent to indicate the efficiency of the resource available. The power grid computing lies in its capacity to aggregate widely distributed disparate resources, and contribute non trivial benefits to users. Task management or workload management is a key issue that must be solved in grid computing and a better scheduling scheme can greatly improve the efficiency of the grid. In order for users to gain simultaneous access for their applications to be accomplished in parallel, prior reservation of resource is necessary. **Results:** The performance of scheduling with load balancing and queuing using TBRA is analyzed and its performance is noticed to be satisfactory in terms of resource utilization, processing time, time consumption and memory usage. **Conclusion:** The proposed architecture aims at establishing a novel approach to balanced unbalanced workloads in a dynamic grid computing environment. It is orchestrated by the various components that are crucial in facilitating dynamic load balancing based on patterns in a more systematic fashion. The design could be furthered by including components that would make the architecture more adaptive. In this model we came across a combination of three widely used mechanisms of scheduling a task— Queuing, Trusted resource and Advanced Reservation. This combination allows a more wide selection of scheduling a task. Our model helps to utilize resources to the maximum level possible and thus give high profit to resource providers. In regular advance reservation procedures we only find one suitable resource out of many resources, but in the proposed model since we sort the resources based on trust factor, we make sure that the user gets a reservation in the best resource possible. Queuing mechanism in our proposed model helps to prioritize tasks before scheduling, in order to provide efficiency. The sorting mechanism of tasks within a queue helps to increase the efficiency thereby producing a very high throughput.

To Cite This Article: Venkatesan. R and Dr. Thanushkodi. K., LB Scheduling for Advanced Reservation and Queuing Using TBRA in Grid Computing Environments. ***Aust. J. Basic & Appl. Sci.,*** *8(3): 179-187, 2014*

INTRODUCTION

THE emergence of grid computing has developed a platform for industrial applications evaluating big quantities of data. Since a grid is a cluster of computers, it integrates a heterogeneous selection of computers which are, probably, dispersed geographically. Such an environment is prone to a set of weakness which incorporates slowness of the access times and unbalanced workloads. To enhance work load, load balancing is required in a grid computing environment. Load balancing is decisive when multiple computers are linked together to share the computational workload or function as a single virtual computer. Dynamic load balancing, ideally, should be designed such that the compromise between extreme resources can be manipulated by the developer. Logically, from the user side they are numerous machines, but act as a single virtual machine. Requests originated from the user are handled by, and distributed among all the stand alone computers to form a cluster. This concludes with balanced computational work among numerous machines, developing the performance of the cluster systems.

Corresponding Author: Venkatesan R., Assistant professor, Karunya University, Department of Information Technology, Venkatesan. R, Box.3030. Coimbatore. India. Karunya University, Karunya Nagar, Coimbatore-641114, Tamil Nadu.
Phone numbers 0422- 2614300; E-mail: rlvenkei2000@gmail.com

Scheduling is a primary technique in computer multitasking, and real time operating system designs. Scheduling indicates to the manner in which tasks are assigned to free CPUs, since there are typically many more tasks than there are available CPUs. The process of choosing a task to be scheduled is done by the scheduler. A scheduling algorithm that is capable of scheduling the right task at the time can produce very high throughput. A computational grid comprises of PCs, workstations, clusters, supercomputers, multi-processor or parallel processor machines. These resources are contributed and shared by various organizations; where each organization can either act as a user or a resource provider. Trust based resource selection is essential in a grid where users and resource are anonymous to each other. When a user wants to access a resource he must be guaranteed for receiving the expected output from the resource provider. A proper queuing strategy must be utilized in order to service the right user at the time.

A grid being a highly dynamic system; predicting the arrival of tasks, it's scheduling and delivery to the user is highly unguaranteed. To overcome this disadvantage advance reservation of resource is necessary. In this paper we propose an algorithm to perform an efficient reservation. Advanced reservation, a technique to support QoS has been incorporated in many grid systems.

Related Work:

There have been several established approaches to balance load dynamically in a grid. Numerous parameters have surfaced in recent times which aid in gauging efficiency and productivity of techniques against set standards and norms.

Kim, et al proposed a paper for optimal job scheduling in grid computing (Kim, S.-S., 2013). A computational grid is a broad scale, heterogeneous selection of autonomous systems, geographically dispersed and interconnected by low-latency and high bandwidth networks. Grid resource management contributes functionality for the discovery and publishing of resources as well as scheduling, submission and monitoring of jobs. *Kokilavani and Amalarethinam* suggested a load balanced Min-Min algorithm for static Meta-Task scheduling in grid (Kokilavani, T. and D.G. Amalarethinam, 2011). Scheduling is considered to be an important issue in the current grid scenario. The demand for effective scheduling increases to achieve high performance computing. Opportunistic Load Balancing assigns the jobs in a random order in the next available resource without considering the execution time of the jobs on those resources. Thus it provides a load balanced schedule but it produces a very poor makespan. Minimum execution time assigns jobs to the resources based on their minimum expected execution time without considering the availability of the resource and its current load. *Xhafa and Abraham* proposed a paper for computational models and heuristic methods for grid scheduling problems (Xhafa, F. and A. Abraham, 2010). For the majority of grid systems, scheduling is a very crucial mechanism. In the simplest cases, scheduling of jobs can be done in a blind way by simply assigning the incoming tasks to the available compatible resources. An important issue here is how to formally define the grid scheduling problem. In this paper they present the most important and useful computational models for this purpose. *Awad and Deering* proposed a paper for an improved constraint based resource scheduling approach using job grouping strategy in grid computing (Awad, I. and J. Deering, 2013).

Scheduling framework for bandwidth aware job grouping based scheduling approach uses the bandwidth in scheduling framework to enhance the performance of job scheduling, but this algorithm does not utilize the resources efficiently. *Liu, et al* proposed a paper for scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm (Liu, H., 2010). Grid resource management provides functionally for the discovery and publishing of resources as well as scheduling, submission and monitoring of jobs. Every resource owner might have a unique way of managing and scheduling resources and the grid schedulers must ensure that they do not conflict with the resource owner's policies. Schedule is the mapping of the tasks to specific time intervals of the grid nodes. A scheduling obstacle is stated by, a set of jobs/operations, a set of machines optimality principle, environmental specifications, and by other constraints. *Yagoubi and Meddaber* proposed a paper for distributed load balancing model for grid computing (Yagoubi, B. and M. Meddeber, 2010). Load balancing algorithms adopting an application centric objective function aim to optimize the performance of each separate function. Most of current grid applications concerns are about time, such as the, makespan. To diminish the time needed to execute all tasks, the workload has to be uniformly dispersed over all nodes which are based on their processing capabilities. This is why load balancing is needed. *Revar, et al* recommended a paper for load balancing in grid environments using machine learning (Revar., A., 2010).

Users are privileged to use grid for solving large computational troubles. It has been heavily demanded to classify the issues affecting the performance. The essential improvement of grid is to contribute resources among numerous applications. Therefore, the amount resources available to any given application highly fluctuate over time. In this scenario load balancing plays key role. In grid environment with efficient load balancing enhancing enhanced system performance and a lower turnaround time for individual jobs can be achieved. *Balasangameshwara and Raju* proposed a paper for a hybrid policy for fault tolerant load balancing in grid computing environments (Balasangameshwara, J. and N. Raju, 2012). Load balancing can be further categorized as static, dynamic or adaptive algorithms based on the type of information on which the load balancing decisions are made. Static algorithms assume that all information including the characteristics of jobs, the resources and communication network are known in advance.

Kamarunisha, et al recommended a paper for recitation of load balancing algorithms in grid computing environments (Kamarunisha, M., 2011). Focus of this paper is on analyzing load balancing requirements in a grid environment and proposing a centralized and sender initiated load balancing algorithm. Grid resource management is defined as the process of identifying requirements, identical resources to operations, scheduling those resources, and scheduling and monitoring grid resources over time in order to run grid applications as efficiently as possible. *Pathak, et al* proposed a paper of an efficient scheduling policy for load balancing model for computational grid system (Pathak, M., 2012). Work load and resource management are two essential functions provided at the service level of the grid system. *Prakash kumar, et al* proposed a paper for backfilling strategies for computational grid system load balancing (Kumar, P., 2013). Load balancing is one of the most important factors which can affect the performance of the grid application. This paper aims to design and development of a performance efficient load balancing algorithm.

Matei, et al proposed a paper for trust based multi agent filtering for increased smart grid security (Matei, I., 2012). In this paper, they combined the multi agent filtering scheme with a trust based mechanism under which agent associates a trust metric to each of its neighbors. In addition, a mechanism for the trust metric update is also introduced, which establishes that agents that diverge noticeably from their expected behavior have their trust value lowered. *Saravanakumar and Prathima* recommended a paper for a novel load balancing algorithm for computational grid (Saravanakumar, E. and G. Prathima, 2010). The grid computing environment is a cooperation of distributed systems where user jobs can be executed on either local or remote computer. The load balancing is done by migrating jobs to the processors, a set of processors to which a processor is directly connected. An algorithm on Arrival (LBA) is proposed for small scale systems (intraGrid). *Mukhopadhyay, et al* proposed an application of existing load balancing algorithms for dynamic, large, heterogeneous distributed systems (Mukhopadhyay, R., 2010).

Local scheduling is the assignment of processor time quantum to task as it is done by every traditional operating system. *Alharbi* proposed a simple scheduling algorithm with load balancing for grid computing (Alharbi, F. and K. Rabigh, 2012). Task scheduling is critical to achieving high performance on grid computing environment. The scope of the scheduling action is to map each task with specific requirements to a capable machine in order to diminish the makespan. *Nandagopal, Uthariaraj* suggested a paper for hierarchical status information exchange scheduling and load balancing for computational grid environments (Nandagopal, M. and R.V. Uthariaraj, 2010). This paper addresses the problem of scheduling and load balancing in a grid architecture where computational resources are dispersed in different administrative domains or clusters which are connected to the grid scheduler by means of heterogeneous communication bandwidths is considered. *El-Zoghdy* proposed a hierarchical load balancing policy for grid computing environment (El-Zoghdy, S.F., 2012). The service level of the grid software infrastructure provides two essential functions for workload and resource management. To conveniently employ the resources at these environments, adequate load balancing and resource management policies are essentially imperative. This paper addresses the obstacle of load balancing and task migration in grid computing environments.

Mehta, *et al.*, 2011: proposed a scheduling algorithms in clusters and grids using improved dynamic load balancing techniques. The large scale distributed computing environment involves a large number of resources providers and resource consumers. Trust management is achieved with the concept of bidirectional reputation points that are assigned to resource providers and resource consumers. *Mana, et al* proposed a paper for a trust negotiation based security framework for service provisioning in load balancing clusters (Maña, A., 2012). The focus of this paper is on the development of an authorization system with integrated trust negotiation capability by emphasizing on its architectural design and negotiation enforcement aspects meeting the security requirements. *Wang, et al* proposed a paper for a physical and virtual compute cluster resource load balancing approach to data parallel scientific workflow scheduling (Wang, J., 2011).

Proposed Methods:

The proposed approach is that of a scheduling framework with resource level load balancing using agents and Queuing with TBRA mechanism in grid computing environments.

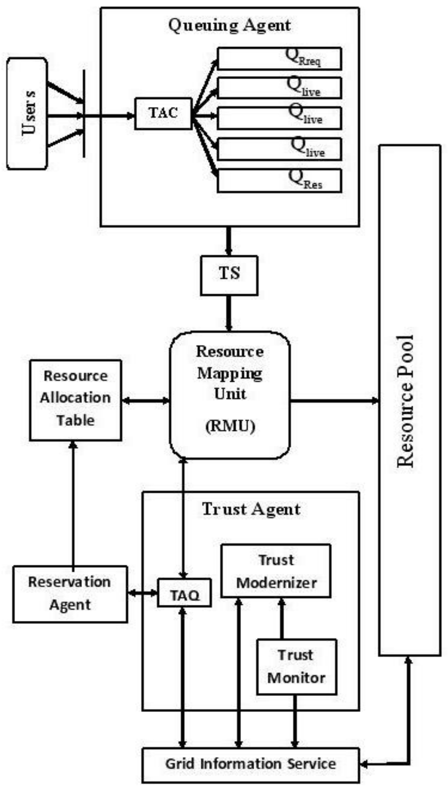
Resource level Load balancing:

The Resource Management Unit (RMU), as the name suggests, is responsible for determining which resources may be allocated to an incoming job. It performs a check on the available resource types prior to any further proceeding within the framework. Load balancing is brought about when there arises a demand for resources which is more that what is available. In such a scenario, job begins to enter a job queue and wait there after pattern mapping for resources to be freed. An information retriever agent relays the details of all the waiting jobs and assumes responsibility to co-ordinate resources accordingly; it also notifies the jobs waiting in the queue when the resources are released. In grid environment, the load of some resources may decreases while the others are overloaded, for the reason that the tasks have different receiving time and properties and the computational resources are heterogeneous. This unbalanced load may cause failure of some tasks on busiest resources, while the others remain unused. Therefore, the resource allocation and load balancing are the main issues, which are needed in resource management for utilizing the grid. Resource scheduling and load balancing are the obstacles in grid environment unit and affect grid performance significantly. On the other hand, the resource reallocation for balancing the load increases the network overload and waiting time of tasks. Therefore, it is so important to use the resource level load balancing technique, which balances the received load through allocation process.

From figure 1 we can understand that there are 3 various types of jobs that are queued. So the job seizure has to prioritize these queues in order to retrieve a job from the queue. Q_{Res} has jobs that have already been reserved, Q_{RReq} has requests that need reservation for future jobs and Q_{live} contains all live jobs distributed across 3 queues based on their delay factor. We cannot delay the jobs in Q_{Res} because any delay in the reserved jobs affect the reservation period of the job. Thus the job seizure checks if Q_{Res} has any jobs and executes them. But if the queue is empty then job seizure checks for Q_{RReq} and reserves jobs if there are any requests for reservation. Only if Q_{Res} and Q_{RReq} are serviced live jobs are serviced.

Queuing System:

Queuing refers to how an incoming user's task is broken down and analyzed; in order to ascertain the target queue and position from which it is scheduled at the presumed time. A centralized scheduler is assumed, i.e. all scheduling mechanisms are done by this single scheduler. Tasks arrive randomly based on Poisson arrival process, and because of this the scheduler cannot wait to receive all incoming tasks but it should schedule tasks as and when a suitable resource is available. Each task received by the scheduler has two key parameters namely, the estimated length of the task and the deadline time of the task. From these two parameters we can derive a numerical value known as the delay factor. The delay factor gives a relative value of how long the task can be delayed before it gets too late to be scheduled and delivered back to the user; therefore 'lower the delay factor higher the priority of the task; higher the delay factor lower the priority of the task'. This shows that delay factor is inversely proportional to the priority of the task.



TAC:	Task Acquaintance
TS:	Task Seizure
TAQ:	Trust Acquaintance
Q_{RReq}:	Queue for reservation request
Q_{live}:	Queue for live jobs
Q_{Res}:	Queue for reserved jobs

Fig. 1: Proposed Framework

For a task t_i, the delay factor df_i can be obtained as,

$$delay\ factor\ (df_i) = \frac{due\ date\ of\ task\ (t_i)}{estimated\ length\ of\ task\ (t_i)} \qquad (1)$$

In this model multiple queues are used; with each queue having certain priority. Now all tasks whose delay factor falls into the lower range are sent to the queue with the highest priority (q_1). Tasks whose delay factor falls into the middle range are sent to the queue with medium priority (q_2). All other tasks are added to the queue with the lowest priority (q_3). Every time a task is added into a queue; all the tasks in that particular queue are sorted based on their delay factor. This sorting within q queue is done so that, all the tasks in that queue are arranged in the proper sequence before being scheduled. Thus it helps to achieve maximum throughput. All the above operation of queuing of a task cannot be done manually by the user. Thus we introduce a task acquaintance component the queue in which it is to be placed.

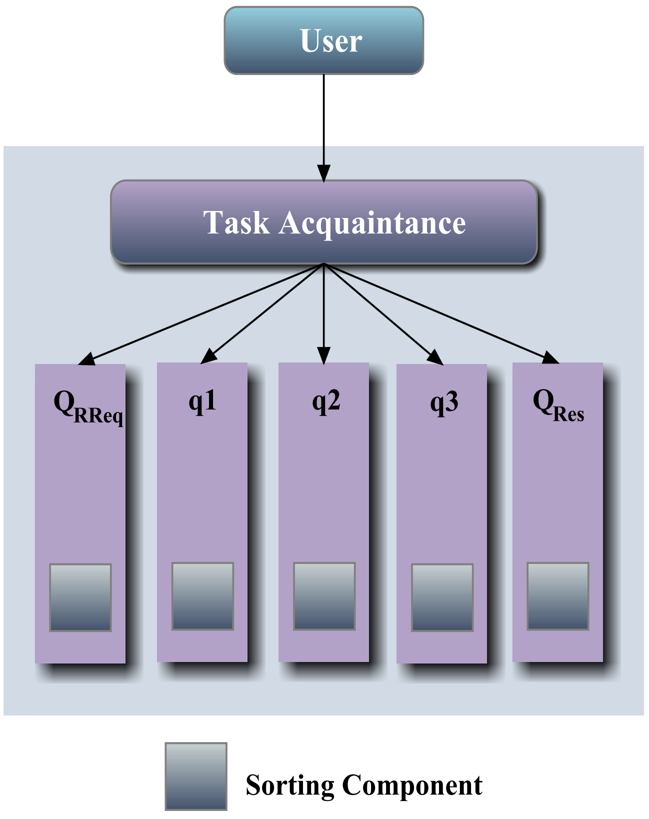


Fig. 2: Queuing System

Trust Based Resource Allocation (TBRA):

Trust is the substantial notion in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behavior and applies only within a specific context at a given time.

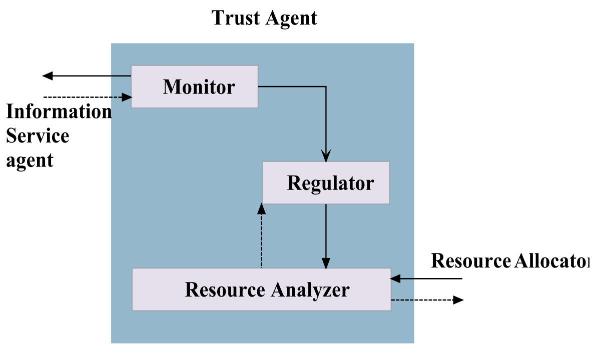


Fig. 3: Trust agent

The information retriever agent (IRA) facilitates communication of data between the resource and the other components. There exist dedicated resources that are segregated in such patterns. This imparts a systematic approach to how load may be balanced in situations where unbalanced workload arises. The trust factor of a resource depends on the total number of tasks run on the resource and the total number tasks completed successfully. Resources with greater trust factor are more reliable than those with lower trust factor. The trust factor tf_a of resource r_a varies from $0 < tf_a \leq 100$.

$$\text{Trust factor } (tf_a) = \frac{\text{Total number of successfull tasks on resource } r_a \times 100}{\text{Total number of tasks on resource } r_a}$$

The role of the Grid Information Service (GIS) is to provide such information to grid schedulers. GIS is important for collecting and predicting the resource state information, such as memory size, software availabilities, network bandwidth, CPU capacities and load of a site in a particular period. A change in the grid information service (GIS) regarding the current state of a resource is monitored by a 'monitor' component.

Advanced Reservation System:

Advanced Reservation is a process of requesting a resource provider for assurance of CPU time for a certain period of time in the future. It allows users to gain concurrent access to adequate resources for applications to be executed. According to our proposed model when a request for reservation comes into the scheduler, the reservation acquisition receives the request and extracts all required information. This information can include start time of reservation, end time of reservation (which can also be derived by start time + length of task), the amount of CPU required and other requirements of the task. This information is sent to resource collection component which finds and collects all resources that can satisfy the task's requirements. Now the collection of satisfying resources is ordered by its trust factor and sent to conflict discovery.

The purpose of the conflict discovery component is to make sure that too many tasks are not scheduled in a single cluster of resource, and thus it helps to prevent failure of reserved tasks. Out of the received set of resources, the resource with maximum trust factor is taken and checked for other tasks reserved within the same time frame of the new incoming task. If there are multiple tasks reserved during the same time frame, we check whether the total amount of resource in the cluster is capable of providing CPU time for all the reserved tasks including the new task. If the resource has enough capacity, we reserve our new task otherwise we try to reserve the new task in the next resource with a lower trust factor. If there are no resources available, then we can either notify the user or we can take a risk of scheduling our new task. In certain cases it is worth taking the risk because usually when an incoming task requests reservation the start time and end time of the requesting task is overestimated; thus causing the resource to be idle for some duration within the reserved period.

Performance And Analysis:

The performance of scheduling with load balancing and queuing using TBRA is analyzed and its performance is noticed to be satisfactory in terms of resource utilization, processing time, time consumption and memory usage.

Processing Time:

The processing time for the number of jobs is minimized by using the Queuing mechanism. TBRA is used in a queuing mechanism in order to diminish the execution time. The processing time of the jobs is analyzed and depicted in fig. 7. The jobs are executed in a queued manner. In this graph, the x-axis represents the jobs to be executed and the y-axis represents the processing time in order to execute the jobs.

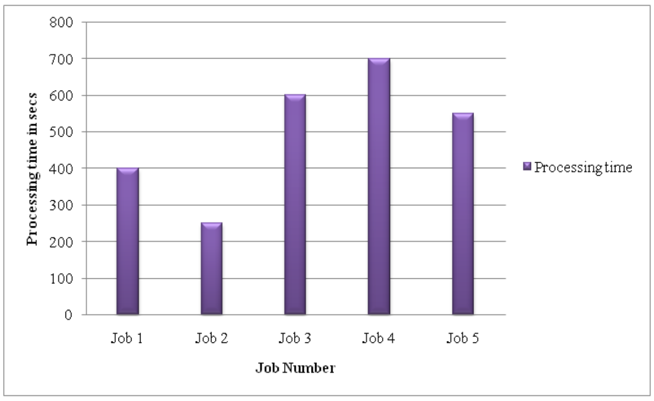


Fig. 4: Processing time

Job Scheduling:

The Resource utilization for the number of jobs is analyzed and showed in fig. 8. Each resource will be allocated to a particular job. The jobs are scheduled and executed based on the load. In this graph, the x-axis represents the schedules and the y-axis represents the resource utilization in percentage.

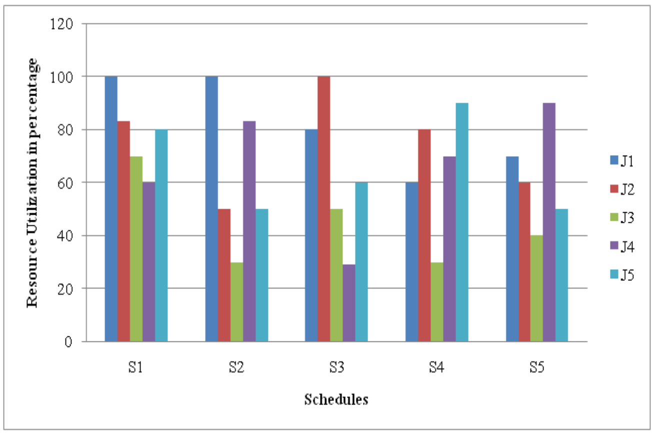


Fig. 5: Job Scheduling

Time Consumption:

The time consumption for the jobs execution are analyzed and depicted in fig.9. In this graph, the x-axis represents the jobs to be executed and the y-axis represents the time in milliseconds.

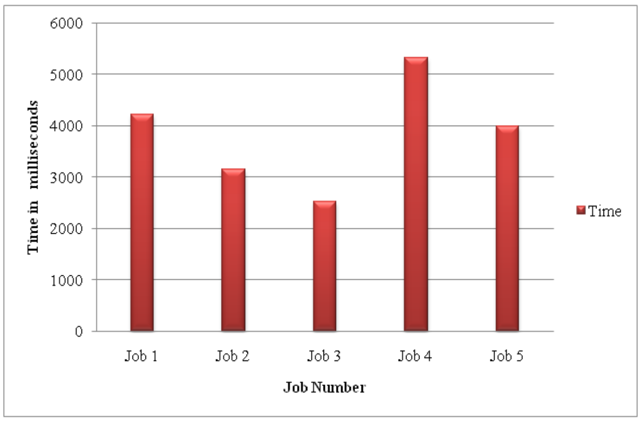


Fig. 6: Time Consumption

Time consumption of job execution will be diminished by using the Queuing mechanism with TBRA.

Memory Usage:

The memory usage for each job is analyzed and showed in fig. 10. In this graph the x-axis represents the jobs to be executed and the y-axis represents the memory in bytes. The usage of memory will be diminished by using the load balancing technique.

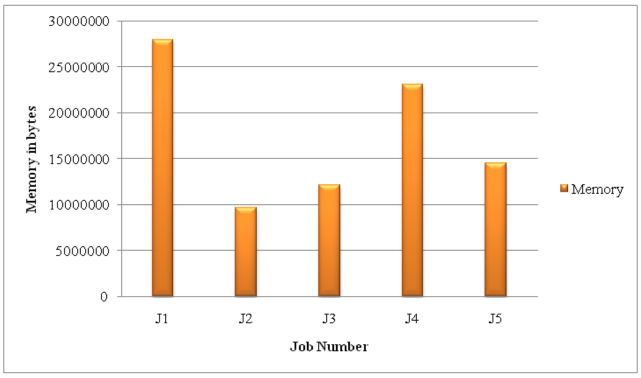


Fig. 7: Memory Usage

Conclusion:

The proposed architecture aims at establishing a novel approach to balanced unbalanced workloads in a dynamic grid computing environment. It is orchestrated by the various components that are crucial in facilitating dynamic load balancing based on patterns in a more systematic fashion. The design could be furthered by including components that would make the architecture more adaptive. In this model we came across a combination of three widely used mechanisms of scheduling a task— Queuing, Trusted resource and Advanced Reservation. This combination allows a more wide selection of scheduling a task.

Our model helps to utilize resources to the maximum level possible and thus give high profit to resource providers. In regular advance reservation procedures we only find one suitable resource out of many resources, but in the proposed model since we sort the resources based on trust factor, we make sure that the user gets a reservation in the best resource possible. Queuing mechanism in our proposed model helps to prioritize tasks before scheduling, in order to provide efficiency. The sorting mechanism of tasks within a queue helps to increase the efficiency thereby producing a very high throughput.

REFERENCES

Alharbi, F. and K. Rabigh, 2012. Simple scheduling algorithm with load balancing for grid computing. Asian Transactions on Computers, 2.

Awad, I. and J. Deering, 2013. An Improved Constraint Based Resource Scheduling Approach Using Job Grouping Strategy in Grid Computing. European Journal of Engineering and Innovation, 11: 1-7.

Balasangameshwara, J. and N. Raju, 2012. A hybrid policy for fault tolerant load balancing in grid computing environments. Journal of Network and Computer Applications, 35: 412-422.

El-Zoghdy, S.F., 2012. A hierarchical load balancing policy for grid computing environment. International Journal of Computer Network and Information Security (IJCNIS), 4: 1.

Kim, S.-S., 2013. Optimal job scheduling in grid computing using efficient binary artificial bee colony optimization. Soft computing, pp: 1-16.

Kokilavani, T. and D.G. Amalarethinam, 2011. Load balanced min-min algorithm for static meta-task scheduling in grid computing. International Journal of Computer Applications, 20: 43-49.

Kamarunisha, M., 2011. Recitation of Load Balancing Algorithms In Grid Computing Environment Using Policies And Strategies An Approach. International Journal of Scientific & Engineering Research, 2.

Kumar, P., 2013. Backfilling Strategies for Computational Grid System Load Balancing.

Liu, H., 2010. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. Future generation computer systems, 26: 1336-1343.

Mehta, H.K., 2011. Performance enhancement of scheduling algorithms in clusters and grids using improved dynamic load balancing techniques. Proceedings of the 20th international conference companion on World wide web, pp: 385-390.

Maña, A., 2012. A trust negotiation based security framework for service provisioning in load-balancing clusters. Computers & Security, 31: 4-25.

Matei, I., 2012. Trust-based multi-agent filtering for increased smart grid security, Control & Automation (MED), Mediterranean Conference., pp: 716-721.

Mukhopadhyay, R., 2010. A study on the application of existing load balancing algorithms for large, dynamic, heterogeneous distributed systems. Proceedings of the 9th WSEAS international conference on Software engineering, parallel and distributed systems, pp: 238-243.

Nandagopal, M. and R.V. Uthariaraj, 2010. Hierarchical Status Information Exchange Scheduling and Load Balancing For Computational Grid Environments. IJCSNS International Journal of Computer Science and Network Security, 10: 177-185.

Pathak, M., 2012. An efficient scheduling policy for load balancing model for computational grid system. Computer Engineering and Intelligent Systems, 3: 51-61.

Revar., A., 2010. Load balancing in grid environment using machine learning-innovative approach. International Journal of Computer Applications (0975–8887), 8: 1.

Saravanakumar, E. and G. Prathima, 2010. A novel load balancing algorithm for computational grid. Innovative Computing Technologies (ICICT), International Conference., pp: 1-6.

Wang, J., 2011. A physical and virtual compute cluster resource load balancing approach to data-parallel scientific workflow scheduling. *Services (SERVICES), IEEE World Congress*, pp 212-215.

Xhafa, F. and A. Abraham, 2010. Computational models and heuristic methods for Grid scheduling problems. Future generation computer systems, 26: 608-621.

Yagoubi, B. and M. Meddeber, 2010. Distributed load balancing model for grid computing. Revue ARIMA, 12: 43-60.