Enhanced Fault Tolerant Based Job Scheduling With Load Balancing Approach in Computational Grid

Dr. P. Suresh[1], Dr. P. Keerthika [2]

[1] Department of Information Technology, Kongu Engineering College, Perundurai, Erode-638052, Tamilnadu, India
[2] Department of CSE, Kongu Engineering College, Perundurai, Erode-638052, Tamilnadu, India.

Abstract— *Grid computing is a process of applying the resources of many computers in a network to a single problem at the same time usually to a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data. Resource management and scheduling are key grid services. The two main techniques that are most suitable to cope with the dynamic nature of the grid are load balancing and job replication. The fault tolerant scheduling algorithm namely MinRC (Minimum Replication cost and Completion time) is developed. It employs peer-to-peer fail over strategy such that each resource is a backup of another neighbour or partner resource in the grid system. The load balancing algorithm is developed to find the load of each resource and also to choose the resource which has minimum load. The fault tolerant scheduling is integrated with load balancing algorithm and developed a Performance Enhanced fault tolerant scheduling algorithm (PE_MinRC). In order to improve system flexibility, reliability and save system resource, fault-tolerant load balancing algorithm employs passive replication scheme. The main objective is to arrive at job assignments that could achieve maximum resource utilization, minimum response time and a well balanced load across all the resources involved in a grid. This approach gives relatively low overhead and robust performance against resource failures and inaccuracies in performance prediction information.*

Keywords— **Fault tolerance, Grid computing, Job scheduling, Load balancing, Replication.**

I. INTRODUCTION

Grid computing discipline involves the actual networking services and connections of a potentially unlimited number of ubiquitous computing devices within a grid. It requires the use of parallel processing software that can divide a program among as many as several thousand computers and restructure the results into a single solution of the problem. It is an infrastructure of resource sharing. It is a technology for using enormous amounts of computing power and data storage, a possibility to share expensive computational resources.

Grid is a parallel and distributed system in which resource spread across multiple administrative domains are able to select, share and integrate based on common rules they accept[1]. It aims ultimately to turn the global network of computers into a vast computational resource. Grid systems are classified into two categories: compute and data grids. In compute grids, the main resource that is being managed by the resource management system is compute cycles (i.e. processors); while in data grids the focus is to manage data distributed over geographical locations. The type of grid system it is deployed in affects the architecture and the services provided by the resource management system. In this paper, we consider fault tolerant scheduling and balancing application load for a computational grid by taking into account grid architecture, computer heterogeneity, resource unpredictability and communication delay.

Fault tolerance is the ability of a system to perform its function correctly even in the presence of faults [2]. A fault tolerant service detects errors and recovers from them without participation of any external agents, such as humans. Errors are detected and corrected and permanent faults are located and removed while the system continues to deliver acceptable services. Strategies to recover from errors include roll-back, which implies bringing the system to a correct state saved before the error occurred, roll forward, i.e. bringing the system to a fresh state without errors, or compensation, i.e. masking an error, in situations when the system contains enough redundancy to do that.

The fault tolerance makes the system more dependable. In a broad sense, fault tolerance is associated with reliability, with successful operation, and with the absence of breakdowns. A fault-tolerant system should be able to handle faults in individual hardware or software components, power failures or other kinds of unexpected disasters and still meet its specification. Fault tolerance is needed because it is practically impossible to build a perfect system. The majority of fault-tolerant designs have been directed toward building computers that automatically recover from random faults occurring in hardware components.

Resource failures (processors/links) may frequently occur in grid systems and have an adverse effect on applications. Consequently, there is an increasing need for developing techniques to achieve fault tolerance [3]. In multiprocessor systems, fault tolerance can be provided by scheduling replicas of jobs on different processors. There are two main approaches to replication as described below.

(1) Active replication: This technique is based on space redundancy i.e. multiple copies of each job are mapped on different processors, which are run in parallel to tolerate a fixed number of failures. With such a technique, no fault detection mechanism is required.

(2) Passive replication: The main idea of this technique is that a backup copy of a job is activated only if a fault occurs while executing its primary copy. It does not require fault diagnosis and is guaranteed to recover all affected jobs by processor failure. In such a scheme, only two copies of the job are scheduled on different processors (space exclusion) and time exclusion. This approach is very useful for grid where fault diagnosis is very difficult.

Load balancing is a computer networking methodology to distribute workload across multiple computers to achieve optimal resource utilization, maximize throughput, minimize response time and to avoid overload. A typical distributed system involves a large number of geographically distributed worker nodes which can be interconnected and effectively utilized in order to achieve performance not ordinarily attainable on a single node [4]. To minimize the time needed to perform all tasks, the workload has to be evenly distributed over all nodes which are based on their processing capabilities. The load balancing problem is closely related to scheduling and resource allocation. It is concerned with all techniques allowing an evenly distribution of the workload among the available resources in a system. The main objective of a load balancing consists primarily to optimize the average response time of applications; this often means the maintenance the workload proportionally equivalent on the whole system resources.

Load balancing algorithms can be classified into two categories: static [5] or dynamic [6]. In static load balancing algorithms allocate the tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based on an average load of our workstation cluster [7]. The decisions related to load balance are made at compile time when resource requirements are estimated. The scheduling is carried out according to a predefined approach. In dynamic load balancing algorithms make changes to the distribution of work among workstations at run-time; they use current or recent load information when making distribution decisions. Multi computers with dynamic load balancing allocate/reallocate resources at runtime based on no a priori task information, which may determine when and whose tasks can be migrated. As a result, dynamic load balancing algorithms can provide a significant improvement in performance over static algorithms [8]. Thus, a dynamic approach can be made adaptive to changes in system parameters such as job arrival rate, CPU processing rate, loads and communication bandwidth between computers.

II. RELATED WORK

The grid environment consists of dynamic and heterogeneous resources. Resource changes with time due to addition of new resource or any of the existing resource leaving the grid environment. Due to uneven job arrival patterns and unequal computing capabilities, some resources in the grid environment get overloaded or some resources get under loaded or some resources remain idle [9]. The occurrence of the resource failures is high due to the resource characteristics. Both resource failures and load balancing degrade the system performance.

Qin Zheng et al [10] proposed an algorithm for scheduling independent jobs. For all processors where the primary is scheduled on, boundary schedules within the time window are considered and replication cost is compared. It does not overlap with any primary schedule or non over-loadable backup schedule. The boundary schedule which has minimum replication cost. The algorithm first considers boundary schedules of the time window. Then, all existing schedules within or overlapping with the time window are examined one by one and their respective boundary schedules are considered.

Balasangameshwara and Raju [11] proposed a fault tolerant scheduling policy to handle failures that can happen in a grid environment. If a resource shuts down manually, it sends a notice message to its backup, neighbour and partner resources before shutting down. If a resource fails suddenly without any notice then this job replication strategy is used. The backup is scheduled for each primary. In case of failure, the identity of the backup and notice message regarding failure of primary is sent to all neighbour and partner resources. If a job successfully completed, the primary sends a release message to the backup it had reserved such that the released slot of backup can be reserved by other primary.

Yang et al [12] proposed a hybrid load balancing policy which integrates static and dynamic load balancing techniques. Essentially, a static load balancing policy is applied to select effective and suitable node sets. This will lower the unbalanced load probability caused by assigning tasks to ineffective nodes. When a node reveals the possible inability to continue providing resources, the dynamic load balancing policy will determine whether the node in question is ineffective to provide load assignment. The system will then obtain a new replacement node within a short time, to maintain system execution performance.

Syed Nasir et al [13] proposed a multilevel hybrid scheduling algorithm. It is based on master/slave architecture. It uses the round robin allocation strategy for job distribution among the slave processors and the hybrid scheduling is used on each slave processor for computation. The main idea is to execute jobs optimally with minimum average waiting, turnaround and response time defined variable. It executes the longest job thus avoiding starvation. It supports true scalability.

Payli et al [14] proposed a dynamic and a distributed protocol. The grid is partitioned into a number of clusters. Each cluster has a coordinator to perform local load balancing decisions and also to communicate with other cluster coordinators across the grid to provide inter-cluster load transfers. The distributed protocol uses the clusters of the grid to perform local load balancing decision within the clusters and if this is not possible, load balancing is performed among the clusters under the control of cluster heads called the coordinators.
Hao et al [15] used a dynamic, distributed load balancing scheme for a grid environment which provides deadline control for tasks. Initially the resources check their state and make a request to the grid broker according to the change of state in load. Then, the grid broker assigns gridlets between resources and scheduling for load balancing under the deadline request. The experimental results shows that this algorithm can reduce the make span, improve the finished rate of the gridlet and reduce the submitted time.

III. PROPOSED MERTHODOLOGY

The fault tolerant scheduling policy called MinRC for independent jobs is designed. It handles failure that can happen in a grid environment. If a resource shuts down manually, it sends a notice message to its backup, neighbour and partner resources before shutting down. If a resource fails suddenly without any notice then this job replication strategy is used. It employs peer-to-peer fail-over strategy such that each resource is a backup of another neighbour or partner resource in the grid system. The backup is scheduled for each primary and they are located on two different resources. MinRC is triggered by the arrival of a job for scheduling to the primary. The primary sends a replica of the job to the designated backup. In case of failure, the identity of the backup and notice message regarding failure of primary is sent to all neighbour and partner resources. If a job successfully completes, the primary sends a release message to the backup it had reserved such that the released slot of backup can be reserved by other primary. PE_MinRC efficiently considers the system reliability and resource utilization.

3.1 Boundary Schedules

Scheduling the backup of job where its start time and finish time collide with boundaries of the interval or boundaries of over-loadable backup schedules is referred to as boundary schedule. Replication cost is defined as the actual percentage of time needed to be scheduled for the backup besides all overloaded periods with existing backups. Replication cost to schedule backup of job j_i on resource c_i is defined as,

$$R_i^R(j) = (t_e(j)) - t^0(j)) / t_e(j) \qquad (1)$$

where,

$R_i^R(j)$	-Replication cost of job j.
$t_e(j)$	- Deadline of job j..
$t^0(j)$	- Amount of time that backup of job j_i can overload with other backups.

3.2 Scheduling Backups

For all neighbour and partner resources besides the one where the primary is scheduled on boundary schedules within the time window are considered and their replication cost is compared. The boundary schedule which has minimum replication cost is chosen

$$R^R(j_i) = \min \{R_i(j_i)\}, 1 \le i \le M \qquad (2)$$

where M represents the neighbour and partner resources for the primary $c_{i.}$ In case a tie happens, the boundary schedule which can complete earliest is selected. A schedule is eligible if it is within the time window and does not overlap with any primary schedule or non-over loadable backup schedule. Once a backup site is assigned to execute the job in case of failure, it will not accept any backup requests from other sites until the time limit expires or the job assigned to its primary site completes. It attempts to schedule backup with minimum replication cost.

3.3. Load balancing

PE_MinRC algorithm is used to balance the load efficiently and also to improve the performance of backups. This work is motivated by the need of efficient algorithms which takes into account grid architecture, computer heterogeneity and communication delay and resource unpredictability. The main objective is to arrive at job assignments that will achieve minimum response time and minimum load difference between heaviest and lightest resource even during resource failure there by resulting in better resource utilization. This mainly focused on designing a dynamic fault tolerant load balancing algorithm.

Load of resource is defined as the total length of the jobs divided by the current resource capacity. Before scheduling the job, expected load of the resource can be calculated by adding the no of the jobs with particular resource capacity.

Load is calculated based on the formula

$$Load_i = \sum_{j=1}^{n} \frac{MI}{\text{Capacity of Ri}} \qquad (3)$$

where,

MI = Length of jobs

When tasks are allocated to resource, current load is calculated and the tasks are transferred to the neighbourhood resource, when resource becomes over loaded. It achieves better resource utilization even during heavy job arrival rates by considering underloaded resource in grid. This is the first to collectively consider performance- enhanced fault tolerant load balancing and grid scheduling with minimum cost. It is used in estimating the system parameters and in proactive job migration.

PE_MinRC manage the backup even during heavy job arrival rates. It is flexible approach in dealing with the changes that happen in the grid and better resource utilization even during resource failures. It aims to minimize the variations in loads on all machines. Efficiency of decision making is highly reliable and moderate communication cost.

IV. EXPERIMENTAL RESULT

GridSim tool is used to evaluate the performance in simulation environment. It is a java based discrete-event toolkit that enables users to model and simulate the characteristics of grid resources and networks with different configurations. In fig.1 cost is considered with the number of jobs (gridlets). The cost value is high in MinRC algorithm due to overloaded resources. But in PE_MinRC, cost is less even during heavy job arrival rates by considering the non overloaded resources. Therefore it results in better utilization and minimum cost. The result is shown below.

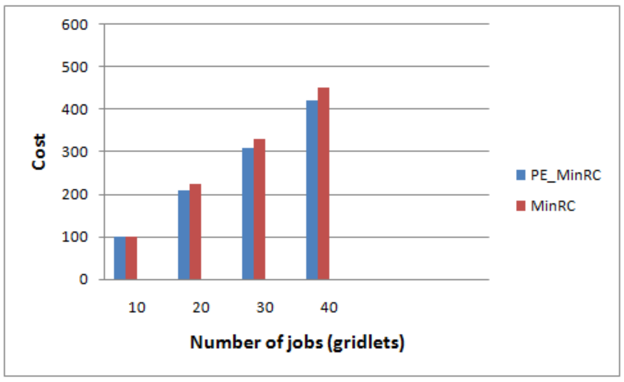

Fig.1 Cost vs. Number of jobs

V. CONCLUSION

Fault tolerance and load balancing are crucial issues for the efficient operation in grid computing environments for distributing the jobs. The designed algorithm considers fault tolerance with minimum replication cost and dynamic load balancing for efficient decisions. It results in better resource utilization even during resource failures. Issues related to security have not been considered in this research. In the future, it is planned to explore the potential of load balancing models by embedding them into real world grid environments.

REFERENCES

[1] Foster, I., Kesselman, C. (eds.), “The Grid: Blueprint for a Future Computing Infrastructure,” 2nd Edition, Morgan Kaufmann, San Mateo 2004.

[2] J.H. Abawajy, “Fault-Tolerant Scheduling Policy for Grid Computing Systems,” Proc. Int’l Parallel and Distributed Processing Symp. (IPDPS), 2004.

[3] Anne Benoit, Mourad Hakem and Yves Robert, “Multi-criteria Scheduling of Precedence Task Graphs on Heterogeneous Platforms,” The computer journal, vol. 53, no. 6, 2010.

[4] Qin Zheng, Chen-Khong Tham, Bharadwaj Veeravalli, “Dynamic Load Balancing and Pricing in Grid Computing with Communication Delay,” J Grid Computing , vol. 6, pp. 239–253, 2008.

[5] Kalim Qureshi, Attiqa Rehman and Paul Manuel, “Enhanced GridSim Architecture with Load Balancing,” Springer Journal of Super Computing, LLC 2010.

[6] Dhakal, S., Hayat, M.M., Pezoa, J.E., Yang, C., Bader, and D.A.:,”Dynamic load balancing in distributed systems in the presence of delays: a regeneration theory approach,” IEEE Trans. Parallel Distrib. Syst., vol. 18, no. 4, pp. 485–497, 2007.

[7] Yun-Han Lee, Seiven Leu, Ruay-Shiung Chang, “Improving Job Scheduling Algorirthms in a Grid Environment,” Future Generation Computer Systems, vol. 27, pp. 991-998, 2011.

[8] Kameda.H. , Li, J., Kim, C., Zhang, Y.,”Optimal Load Balancing in Distributed Computer Systems,” Springer, London ,1997.

[9] Karimi, Faraneh Zarafshan, Adznan.B. Jantan, A.R Ramli, M.Iqbal, B.Saripan, “A New Fuzzy Approach for Dynamic Load Balancing Algorithm,” IJCSIS, vol. 6, no. 1, pp 01-05, 2009.

[10] Qin Zheng, Bharadwaj Veeravalli and Chen-Khong Tham, “On the Design of Fault-Tolerant Scheduling Strategies Using Primary-Backup Approach for Computational Grids with Low Replication Costs”, IEEE Transactions on Computers, vol.58, no.3, 2009.

[11] Balasangameshwara.J and Raju.N, “Performance - Driven Load balancing with Primary-Backup Approach for Computational Grids with Low Communication Cost”, IEEE Transactions on Computers, Vol.35, 2012.

[12] Li.Y, Yang.Y and Zhou, “A hybrid load balancing strategy of sequential tasks for grid computing Environments”, Future Generation Computer Systems, Vol.25, pp. 819-828,2009.

[13] Syed Nasir Mehmood Shah, Ahmad Kamil Bin Mahmood, Alan Qxley, “Dynamic Multilevel Hybrid Scheduling Algorithms for Grid Computing”, Future Generation Computer Systems,Vol.27, pp. 1035-1046, 2011.

[14] Payli R.U, Erciyes.K and Dagdeviren.O, “Cluster - Based Load Balancing Algorithms for Grids”, International Journal of Computer Networks & Communications (IJCNC), Vol.3, No.5, pp. 253-269, 2011.

[15] Li Y, Lan Z, “ A survey of load balancing in grid computing”, In: Lecture notes in computer science, vol. 3314. Springer, Berlin, Heidelberg, pp 280–285, 2005.