An Adaptive Job Scheduling with efficient Fault Tolerance Strategy in Computational Grid

S. Gokuldev [#1], R. Radhakrishnan [*2]

[#] Department of CSE, SNS College of Engineering, Coimbatore, TamilNadu, India
[1] gokuldevs@gmail.com
[*] Sri Shakthi Institute of Engineering and Technology, Coimbatore, TamilNadu, India
[2] radkrishr@yahoo.com

Abstract—Grid computing is an emerging technology which has the potential to solve large scale scientific problems in an integrated heterogeneous environment. However, in the grid computing environment there are certain aspects which reduces efficiency of the system. Scheduling the jobs to the best suited resources, achieving the load balancing and fault tolerance are the key aspects to improve the efficiency and to exploit the capabilities of emergent computational systems. Because of dynamic and distributed nature of the grid, the traditional methodologies of scheduling are inefficient for the effective utilization of the available resources. In this paper, an efficient adaptive job scheduling algorithm is proposed to improve the efficiency of the grid system for a large number of tasks. Moreover, the proposed adaptive job scheduling in addition to the fault tolerance strategy with check pointing approach shows the improvement in performance of the overall computation time even in worst scenario under the heterogeneous grid environment. The simulation results illustrates that the proposed strategy effectively schedules the grid jobs with more than 10% increase in overall performance thus resulting in minimization of overall execution time.

Keyword-Checkpoint, Computational Grid, Fault Tolerance, Job Scheduler, Resource Monitor

I. INTRODUCTION

The improved popularity of the Internet and the availability of powerful computers forms high-speed networks at low-cost has emerged various technologies for their effective utilization of the resources. In the mid-1990s by exploring the design and development of an analogous infrastructure [1] tossed a new technology called grid computing. Grid is a type of parallel and distributed systems that enables effective sharing of resources at run-time based on their efficiency, capability and performance. The selection and aggregation of distributed autonomous resources with respect to the incoming load have also been performed dynamically in the runtime environment. The geographical coupling of the required resources solves very large scale problems which improves the QoS in the grid environment. The resource sharing is one of the significant roles which extracted the view of grid computing, where there are many Virtual Organizations (VO) [2] interconnected with various forms of resources and services and hence the grid resources are of highly dynamic and unstable due to its distributed architecture.

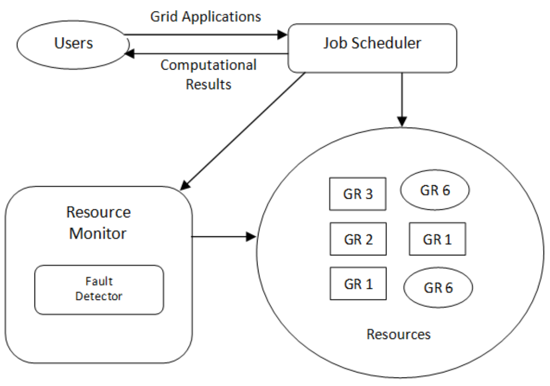


Fig 1. Grid Infrastructure

Generally there are wide ranges of heterogeneous [3] and geometrically distributed resources in the grid such as the computational resources, storage resources and hardware resources. To achieve the high potential in the computational grid, an effective and highly efficient scheduling strategy is fundamentally important. A grid environment as shown in fig.1 becomes more efficient and powerful only when there is a composition of resources in large set and serves a community. The capacity of the resources varies from each other and the jobs to the resources arrive dynamically. The objective of this work is not limited only towards utilizing all the idle resources to achieve the load balancing [4] but also to propose an efficient job scheduling strategy to enhance the performance of the grid systems. This view emphasis the fault tolerance mechanisms to be included with the system, which imposes various types of fault tolerance techniques [5] [6].

Sending minimal number of jobs to the available resources which is capable of high processing efficiency will tend to an economical suppression of the resource selection and aggregation of distributed autonomous resources. By reducing the transmission time and increasing the CPU utilization, the overall efficiency of the grid system are improved.

Thus along with this introductory part few statistical surveys have been carried out towards representing few existing works in chapter 2. In chapter 3 the implementation of the proposed adaptive job scheduling strategy is covered. The concept of fault tolerance which involves fault identification and rectification strategies has been addressed in chapter 4. The experimental results obtained in the simulation environment are addressed in chapter 5 and chapter 6 includes the conclusion and future work.

II. Related Work

Due to the distributed and dynamic nature of grid [7] the probability of failure is found to be very high and hence fault tolerance services [8][6] has become more crucial area in computational grid. The “Grid problem,” can be defined as, highly secured, more flexible with coordinated resource sharing among dynamic collections of individuals, institutions and resources which are referred to as virtual organization [2] which is a setup with an unique authentication, authorization, providing access to the resource, discovering the type of the resource and few other challenges which may probably be encountered. These classes of problem are addressed by grid technologies [7][9].

In an adaptive [10] fine grained job scheduling algorithm [11] is an approach which focuses on light weighted jobs scheduling strategies. In addition to scheduling the tasks [12], the work also focuses on the resource monitoring methodologies [13] with fault tolerance mechanism [14] in grid computing environment .

Scheduling is one of the core steps which enables to exploit the capability of emerging computational systems such as grid. The methods adopted in traditional systems could not be more appropriate for carrying out effective scheduling of jobs to grid so as to show a better performance in minimizing the execution time and therefore there is a need to propose a better, more efficient and effective scheduling algorithm.

One of the work represents a discrete Particle Swarm Optimization (PSO) approach [15] for grid job scheduling. Grid resources are registered within one or more Grid Information Servers (GIS’s). The end users submit their requests to the Resource Scheduler (RS). Different requests demand for different requirements and moreover the available resources have different capabilities. RS discovers proper resources for executing these requests by querying in GIS and then schedules the tasks to the discovered resources.

The probability of a failure in large-scale grids is much greater than traditional parallel systems as represented in RFOH [16] strategy, the grid system should also be able to identify and manage faults [17] which supports reliable execution of jobs.

Grid system failure handling techniques are classified as task-level and workflow-level [18]. The job scheduling strategies has no matter with fault tolerance, but to improve the overall efficiency of grid system and to reduce the total time taken for overall execution of all jobs scheduled, fault tolerance also plays a vital role in the grid environment. By avoiding assigning jobs to the faulty resource in the grid, it is possible to achieve better performance resulting in minimizing the overall execution time by assigning tasks to an efficient resources. By implementing this mechanism the overall execution time will be reduced sparely. Moreover there exists certain restriction in resource usage, as the access permission of the resources need to be provided by administrators of the private sectors who owns the network resources. In few cases, the grid resources may be permitted to be utilized for certain period of time. However in all different scenarios every resources should be managed and utilized efficiently in this dynamic environment.

Grid jobs are executed by the grid system are as follows:

- Grid users submit their jobs to the grid scheduler along with their QoS requirements specification, i.e., deadline with in which the users wants their job to be executed, type of operating system required etc.
- Grid Scheduler (GS) receives the jobs and immediately schedules those jobs to the best available grid resource by optimizing the time. this is done by referring the free pool of resource availability.
- Results of the jobs executed are submitted back to the users upon successful completion of the jobs.

However the computational grid environment has its own drawbacks:

- If a fault occurs on any of the grid resources, the job is rescheduled to any other suitable available resource which eventually results in failing to satisfy the user's QoS requirement i.e deadline. The reason is simple, as the job is re-executed, it consumes more time.
- In the computational grid, there are certain resources that fulfil the criterion of deadline constraint, but they have a tendency towards faults. In such a scenario, the grid scheduler goes ahead to select the same resource for the mere reason that the grid resource promises to meet user's requirements of the grid jobs and the user needs to get compromised on the QoS parameters of their jobs.

III. PROPOSED SCHEDULING MECHANISM

In grid systems, computing the jobs are measured interms of million instructions per second (MIPS) and a threshold value Mmax is generated on evaluating an average computational capability and the threshold value is fixed.

$$\text{MIPS} < \text{Mmax} \tag{1}$$

i.e., if the MIPS of any job is less than the fixed threshold Mmax as in Eq. 1, then that job is considered as a fine grained job else it is treated as a normal type of job. Based on the type, the jobs will be scheduled by an efficient appropriate job scheduling algorithm.

Due to the highly dynamic and distributed nature of the grid environment, the allocation strategy of the resources more challenging in nature. The strategies for grouping the resources are based on this characteristic. In general there are two methods of resource grouping strategies. The first type consists of a interactive set of services, without a resource manager and the second type is the one in which the grid resource monitoring component [13] have been incorporated which holds the information about the current availability of free resources in the resource pool and the capacity of those resources as shown in fig. 2.

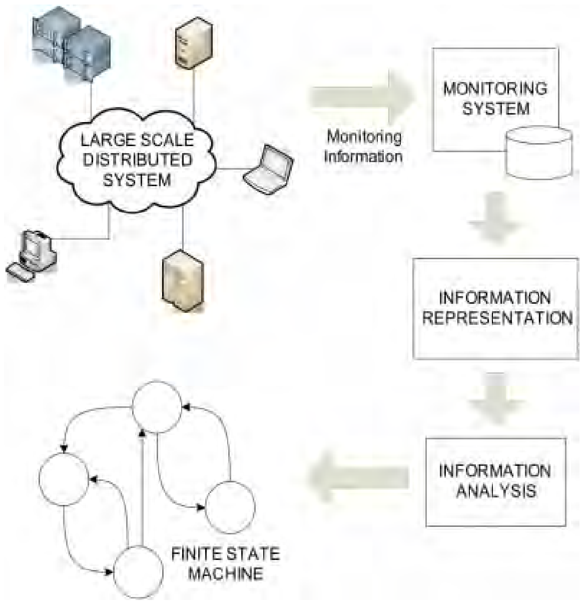


Fig. 1. Scheduling Mechanism by Monitoring System

The information stored in the grid resource monitoring has a mechanism of updating its own information periodically for a given instance of time and also whenever a heavily loaded jobs enters in to the resource pool as scheduled by the resource scheduler. This mechanism is based on the GRIM prototype and GRIR protocol and the resource characteristics on the grouping strategies which are based on the mediators.

$$\Sigma\ C_{i,j} + W_i \tag{2}$$

$$\text{Maxspan} = \text{Max}\ \{\Sigma\ C_{i,j} + W_i\},\ i=\{1.2\ldots m\} \tag{3}$$

$$\text{Flow Time} = \Sigma_{i=1}\ \Sigma\ C_{i,j} \tag{4}$$

where 'i' denotes the resources, 'j' denotes jobs and 'm' denotes previous workload.

According to the above mentioned equations Eq. 3 and Eq. 4, the Maxspan and Flow Time of the jobs scheduled to resources are calculated along with its execution time. Thus, as mentioned, the goal of the resource scheduler is achieved by resulting in reduced make span and flow time rapidly.

IV. FAULT TOLERANCE

Fault tolerance [14] is an assumption that there is a specification of what constitutes correct behaviour of resources. A failure occurs when an actual running system such as in the grid environment deviates from this specified behaviour. The cause of a failure is termed as an error. An error represents an invalid system state, one that is not allowed by the system behaviour specification. The error itself is the result of a defect in the system or fault. In other words, a fault is the root cause of a failure which means that an error is merely the symptom of a fault. A fault may not necessarily result in an error, but the same fault may result in multiple errors in later-on stages. Similarly, a single error may lead to multiple failures.

A. Fault Detection

Detection of faults [17] is limited to the types of faults that a system may handle. The potentially exhaustive list of faults generally avoids the inclusion of arbitrary or Byzantine faults, which simplify the system design. The remaining send omission, crash, and timing failures may be detected by processes devoted to their detection, such as network monitors, or may be detected by any member of the group. It is the responsibility of the monitoring process to notify the group members so that a decision can be made. In the case of crash failures, a crashed process may be detected when the process fails to respond to a message within a timeout period. The sending process then has the responsibility to notify all other members of the group that the process failed.

B. Fault Rectification

The key aspect of the fault rectification in this work is based on implementing the check point mechanism [19] [20]. This mechanism introduces two components namely check point manger and the check point server. The focus of the check point manager is to manage the rate of failure and the total number of failures occurred in the system as such in the computational grid environment.

1) *Checkpoint Manager:* It receives the scheduled tasks from the scheduler and sets checkpoints dynamically [21] based on the failure rate of the resource on which it is scheduled. Then it submits the job to the resource. Checkpoint manager receives job completion message or job failure message from the grid resource and responds to that accordingly. During execution, if job failure occurs, the job is rescheduled from the last checkpoint instead of running from the scratch. Checkpoint manager implements checkpoint setter algorithm to set job checkpoints.

2) Checkpoint Server: On each checkpoint set by the checkpoint manager, job status is reported to the checkpoint server. Checkpoint server saves the job status and returns it on demand i.e., during job or resources failure. For a particular job, the checkpoint server discards the result of the previous checkpoint when a new value of checkpoint result is received.

V. Experimental Results

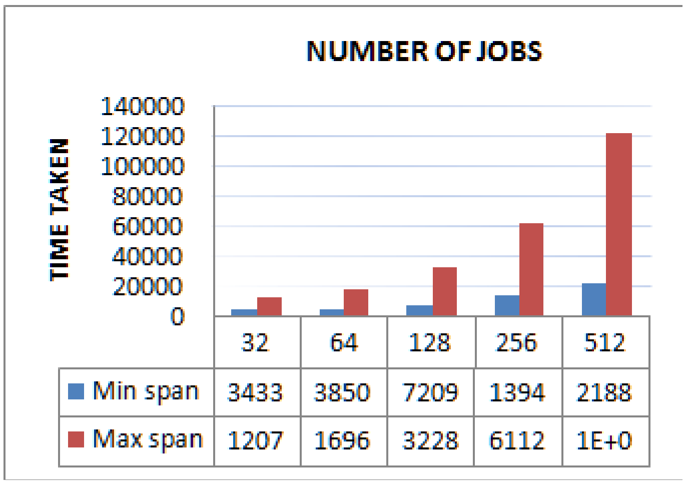
The entire setup of the grid environment and its implementation is carried out in a simulation [22] environment using Gridsim tool [23] and the simulation results are show in table 1. The tool box allows the entire modelling of the distributed system users, applications, resource brokers, resources and fault detector.

TABLE I
Variation in Execution Time

S.No	No. of Tasks	Minspan	Maxspan
1	512	21889	122937
2	256	13944	61129
3	128	7209	32282
4	64	3850	16963
5	32	3433	12078

In this strategy the total number of grid tasks assigned to the grid system varies between a minimum of 2 tasks to a maximum of 512 tasks. The results of the grid depends on the type of tasks allocated to the grid system. As the resources also have certain criteria which are to be considered, the type of resource required can be changed as per the requirement of the user. The number of grid resources for the simulation is assumed as 16 and is changeable based on the system requirements. The execution time and the overall job completion time is based on the type of task that are allocated to the grid system. Tasks varying from 2 to 512 have been executed and verified as there is no change in the total number of available resources and is fixed. The task completion time varies with respect to Minspan to Maxspan as represented in table 1. In the experimental results of the proposed system the Minspan is much lesser than the Maxspan for all the cases. It is clearly evident that whenever there is a gradual decrease in the number of tasks assigned then there is a decrease in the total execution time of jobs with a vast variation in the results obtained for Minspan and Maxspan.

The statistical analysis obtained from experimental results upon executing the proposed algorithm is concluded that, higher the input, higher the efficiency, lower the input lowers the efficiency. The results are executed for varying inputs ranging between 2 to 512, the corresponding results have been obtained for which the total number of resources is set as 16 in this simulation work. The results of inputs 2, 4, 8, 16, are very minimum are considered to be negligible.



	32	64	128	256	512
🟦 Min span	3433	3850	7209	1394	2188
🟥 Max span	1207	1696	3228	6112	1E+0

Fig. 3 Execution time with task variants.

Rest of the values have been executed which proves the exact variations of Maxspan and Minspan algorithms as shown in fig. 3. Hence the result proves that the proposed adaptive job scheduling algorithm is capable of handling larger number of tasks which inturn results in minimizing the overall execution time.

VI. Conclusion and Future Work

The focus of this paper is to research on job scheduling in an heterogeneous grid computing environment and to provide an efficient job scheduling algorithm with an effective fault tolerance strategy. Though there are many job scheduling algorithm in grid computing, most of them concentrates on fine grained job scheduling and few other focuses on only light weighted job scheduling strategies. In case of larger set of resources and more number of tasks, only limited number of scheduling algorithms results good in handling situations. Also, very few algorithms are capable of handling more number of jobs at the same time, along with certain variants of local jobs first and non-local jobs later. The proposed adaptive job scheduling algorithm is one of those which resulted in improved performance.

The introduction of the fault detector and the fault rectifier into proposed system elevated the system with more than 10% of improvement of global performance level of the grid system interms of computation time by satisfying the QoS metric. The adaptive job scheduling algorithm also states that the job scheduling in a dynamic and unstable grid environment is tolerable which reduces the total execution time taken to complete all the jobs assigned in the grid.

The future work is to include certain variants of fault tolerance approaches even in some worst scenarios with improvement in overall execution time.

References

[1] Foster I, Kesselman C (eds.). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann: San Fransisco, CA, 1999.
[2] Ian Foster, C. Kesselman, and S. Tueke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," Supercomputing Applications, 2001
[3] Ekemecic I, Tartalja I, Milutinovic V. A Survey of Heterogeneous Computing: Concepts and Systems. Proceedings of the IEEE 1996; 84(8):1127–1144.
[4] Li. Y, Yang. Y and Zhu. R, "A Hybrid Load balancing Strategy of Sequential Tasks for Computational Grids," IEEE. International Conference on Networking and Digital Society, DOI 10.1109/ICNDS.2009.
[5] Malarvizhi Nandagopal, V. Rhymend Uthariaraj "Fault Tolerant Scheduling Strategy for Computational Grid Environment" International Journal of Engineering Science and Technology vol. 2(9), 2010, pp. 4361-4372.
[6] Ritu Garg and Awadhesh Kumar Singh "Fault Tolerance in Grid Computing: State of the Art and open issues," International journal of Computer Science & Engineering Survey, Vol. 2, No 1, February 2011.
[7] Casavant TL, Kuhl JG. "A Taxonomy of Scheduling in General-purpose Distributed Computing Systems," IEEE Transactions on Software Engineering, 14(2), pp.141–154, 1988.
[8] HwaMin Lee, KwangSik Chung, SungHo Chin, JongHyuk Lee, DaeWon Lee "A Resource Management and Fault Tolerance Services in Grid Computing", Journal of Parallel and Distributed Computing, Vol. 65, pp. 1305-1317, 2005.
[9] Fangpeng Dong and Selim G. Akl "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems," Technical Report No.2006-504.
[10] Gao. Y, Rong. H and Huang. J. Z, "Adaptive Grid Job Scheduling with Genetic Algorithms," Future Generation Computer Systems, vol.21, pp.151-161, January 2005.
[11] Yeqing Liao and Quan Liu, "Research on Fine-grained Job scheduling in Grid Computing" International Journal of Information Engineering and Electronic Business, 2009, 1, 9-16, October 2009.
[12] Kamali gupta, Manpreet singh, "Heuristic based Task Scheduling in Grid", International Journal of Engineering and Technology (IJET), ISSN: 0975-4024, vol. 4, No. 4, Aug-Sep 2012.
[13] Liang Hu, Xi-Long Che and Si-Qing Zheng "Online System for Grid Resource Monitoring and Machine Learning-Based Prediction" IEEE Transactions on Parallel and Distributed Systems Vol. 23, 2012
[14] Latchoumy.P, Sheik Abdul Khader. P "Survey on Fault tolerance in Grid Computing," International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.4, November 2011.

[15] Kennedy J and R .C. Eberhart ‘Particle Swarm Optimization’ Proc. IEEE Int’l Conf. Neural Networks, 1995, pp. 1942-1948
[16] Leyli Mohammad Khanli, Maryam Etminan Far, Ali Ghaffari, ”Reliable Job Scheduler using RFOH in Grid Computing“ Journal of Emerging Trends in Computing and Information Sciences, ISSN 2079-8407 Volume 1, July 2010.
[17] Amoon “ A Fault Tolerant Scheduling System Based on Check pointing for Computational Grids,” International Journal of Advanced Science and Technology,Vol. 48, November, 2012.
[18] Hwang. S and C. Kesselman. “Grid Workflow: A Flexible Failure Handling Framework for the Grid,” 12th IEEE International Symposium on High Performance Distributed Computing (HPDC’03), Seattle, Washington, USA, IEEE CS Press, Los Alamitos, CA, USA, June 22 - 24, 2003.
[19] Maria Chtepen, Filip H.A. Claeys and Bart Dhoedt “Adaptive Task Check pointing and Replication: Toward Efficient Fault-Tolerant Grids,” IEEE Transaction on Parallel and Distributed Systems, Vol. 20, No. 2, February 2009.
[20] Pankaj gupta “Grid computing and checkpoint approach, ”IJCSMS International Journal of Computer Science & Management Studies, VOL. 11, Issue 01, May 2011 ISSN (Online): 2231– 5268.
[21] Antony Lidya Therasa.S, Antony Dalya.S and Sumathi.G “Dynamic Adaptation of Checkpoints and Rescheduling in Grid computing,” International Journal of Computer Application, VOL.3, May 2010.
[22] GridSim. A Grid Simulation Toolkit for Resource Modelling and Application Scheduling for Parallel and Distributed Computing. www.buyya.com/gridsim/.
[23] Buyya. R and Murshed, “Gridsim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing,” Concurrency and Computation: Practice and Experience, vol. 14, pp. 1175–1220, 2002