Global Journal of Advanced Engineering Technologies and Sciences

AN IMPROVED FAULT TOLERANT SCHEDULING ALGORITHM FOR GRID ENVIRONMENT

[1]Keerthika P, [2]Suresh P
[1]Assistant Professor (Senior Grade), Department of CSE,
[2]Assistant Professor (Senior Grade), Department of IT,
Kongu Engineering College, Perundurai, Erode, Tamilnadu, India
keerthikame@gmail.com

Abstract

Grid computing is becoming popular due to the stupendous growth in the internet, bringing the resources together which makes the user to get fast solutions for the large scale problems. This enables sharing, selection and aggregation of a wide variety of geographically distributed resources. As size of the applications grow, more usage of resources for longer periods of time, may lead to increasing number of resource failures. When failures occur, the execution of the jobs that is assigned to the failed resources will be affected. So, fault tolerance is necessary in such cases. To overcome the failure, a scheduling algorithm is proposed that depends on a new factor called scheduling indicator in selecting the resources. This factor comprises of the response time and the fault rate of grid resources. The fault rate is based on the success and the failure of job execution. Whenever a grid scheduler has jobs to schedule on grid resources, it uses the scheduling indicator to generate the scheduling decisions. The resource with minimum scheduling indicator value receives the job.

I. INTRODUCTION

In today's pervasive world, the explosive grid computing environments have become significant that they are often referred to be the world's single and most powerful computer solutions. Previously, the resources were available worldwide in every system. Due to the massive growth of Internet and advent of grid computing, the resources are brought together to make the user, get fast solutions for their jobs. Grid computing is defined as the controlled and coordinated resource sharing and problem solving in dynamic, multi - institutional virtual organizations. It involves the actual networking services and connections of a potentially unlimited number of computing devices within a grid. Grid computing strives for an ideal Central Processing Unit (CPU) cycles and storage of millions of systems across worldwide users [1]. This enables sharing, selection and aggregation of a wide variety of geographically distributed resources. This includes supercomputers, storage systems, data sources and specialized devices used for solving large scale resource intensive problems in science, engineering and commerce. These problems require a great number of computer processing cycles or the need to process large amount of data. The size of a grid may vary from being small, confined to a network of computer workstations within a corporation to a worldwide network.

Computational grids are the solution for all these problems. They offer a convenient way to connect many devices (e.g., processors, memory and Input & Output (I/O) - devices) so that end users are able to combine the computational power of all devices for a certain amount of time. For example, if a user needs to make some CPU consuming calculations, the user could occasionally borrow CPU-time from a grid with a much lower cost than borrow the time from a super computer. A grid could be created in all environments where end users have a computer with memory and CPU. Data grid provides an infrastructure to support data storage, data discovery, data handling, data publication and data manipulation of large volume of data actually stored in various heterogeneous databases and file systems [2]. It deals with the controlled sharing and management of large amounts of distributed data.

Grid scheduling is defined as the process of making scheduling decisions involving resources over multiple administrative domains. The scheduling system must consider the scheduling of jobs involving the mapping of 'n' jobs to 'm' resources. Scheduling is done by using software called job scheduler. Complexity of grids originates from strong variations in the grid availability and an increase in the probability of resources to fail than traditional parallel and distributed systems.

As applications grow, more usage of resources for longer periods of time may lead to increase in number of resource failures. When failures occur, the execution of the jobs assigned to the failed resources will be affected. So, a fault tolerant service

is important in grid environment. Fault tolerance is the ability to preserve the delivery of expected services despite failures within the grid itself. Faults occur when a grid resource is unable to complete the assigned job. Faults occur due to resource failure, job failure or network failure. The resource failure is considered in this work. The jobs submitted by the user are executed by the computational grid by allocating them to the resources with Quality of Service (QoS) requirements. Fig.1 shows the basic grid scheduling model.

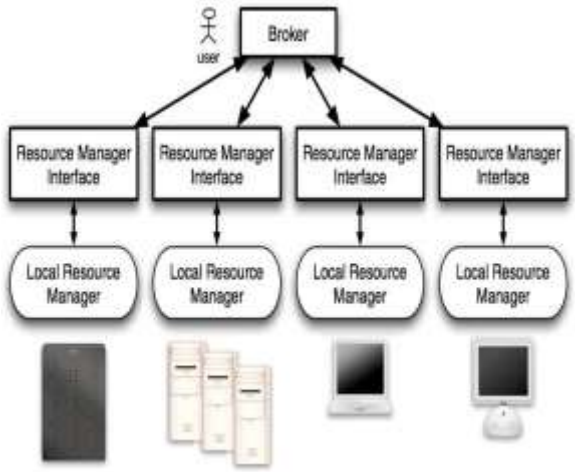

Fig.1 Grid Scheduling Model

A centralized broker is the single point for the whole infrastructure and manages directly the resource manager interfaces that interact directly with the local resource managers. All the users submit the tasks to the centralized broker. Each resource differs from other resources by many ways that includes number of processing elements, processing speed, internal scheduling policy and its load factor etc. Similarly each job differs from other jobs by execution time, deadline, time zone etc.

Fault tolerant mechanisms are needed to hide the occurrence of faults, or the sudden unavailability of resources. Although scheduling and fault tolerance have been traditionally considered independently from each other, there is a strong correlation between them. As a matter of fact, each time a fault-tolerance action must be performed.

II. RELATED WORKS

Fault tolerant scheduling is an important issue for computational grid systems, as grids typically consist of strongly varying and geographically distributed resources.

The problems by combining the checkpoint replication with Minimum Time To Release (MTTR) job scheduling algorithm and fault index is addressed by [4]. Time to release includes the service time of the job, waiting time in the queue, transfer time of input and output data to and from the resource. MTTR algorithm minimizes the time to release by selecting a computational resource based on job requirements, job characteristics and hardware features of the resources. When making scheduling decisions, the scheduler uses the fault index and the response time of resources. It sets the job checkpoints based on the resource failure rate. A critical aspect for an automatic recovery is the availability of checkpoint files. A strategy to increase the availability of checkpoints is replication.

An adaptive fault tolerant job scheduling strategy for economy based grids is proposed in [5]. The proposed strategy maintains a fault index of grid resources. The fault tolerant schedule manager maintains fault history about grid resources and updates Fault Index (FI) of a grid resource by receiving requests from the broker. It dynamically updates the fault index based on successful or unsuccessful completion of an assigned task. Whenever a grid resource broker has tasks to schedule on grid resources, it makes use of the fault index from the fault tolerant schedule manager in addition to using a time optimization heuristic. FI is not a suitable indicator to represent the resource failure history as it cannot be decremented below a certain limit. FI of a grid resource is incremented every time the resource does not complete the assigned job and is decremented whenever the resource completes the assigned job successfully. If the fault index is zero, then the resource with the minimum response time is selected regardless of its failure history. This results in selecting resources that may have higher tendency to fail.

A Failure Detection Service (FDS) mechanism and a flexible failure handling framework is proposed in [6]. The FDS enables the detection of both task crashes and user-defined exceptions. The Grid-WFS is built on top of FDS, which allows users to achieve failure recovery in a variety of ways depending on the requirements and constraints of their applications. The resources are modeled based on the system reliability. Reliability of a grid computing resource is measured by mean time to failure (MTTF), the average time that the grid resource operates without failure. Mean time to repair (MTTR) is the average time it takes to repair the Grid computing resource after failure. The MTTR measures the downtime of the computing resource.

Various fault recovery mechanisms such as checkpointing, replication and rescheduling are discussed in [7]. Taking checkpoints is the process

of periodically saving the state of a running process to durable storage. This allows a process that fails to be restarted from the point its state was last saved, or its checkpoint on a different resource. Replication: Replication means maintaining a sufficient number of replicas, or copies, of a process executing in parallel on different resources so that at least one replica succeeds.

In [8], it is described that the fault tolerance is an important property in order to achieve reliability. Reliability indicates that a system can run continuously without failure. A highly reliable system is the one that continues to work without any interruption over a relatively long period of time. The fault tolerance is closely related to Mean Time to Failure (MTTF) and Mean Time between Failures (MTBF). MTTF is the average time the system operates until a failure occurs, whereas the MTBF is the average time between two consecutive failures. The difference between the two is due to the time needed to repair the system following the first failure. Denoting the Mean Time to Repair by MTTR, the MBTF can be obtained as MTBF=MTTF + MTTR.

A check pointing mechanism is proposed in [9] to achieve fault tolerance. The check pointing process periodically saves the state of a process running on a computing resource so that, in the event of resource failure, it can resume on a different resource. If any resource failure happens, it invokes the necessary replicas in order to meet the user application reliability requirements.

In our previous work [10], we have proposed an efficient fault tolerant scheduling algorithm (FTMM) which is based on data transfer time and failure rate. System performance is also achieved by reducing the idle time of the resources and distributing the unmapped tasks equally among the available resources. A scheduling strategy that considers user deadline and communication time for data intensive tasks with reduced makespan, high hit rate and reduced communication overhead is introduced in [11]. This strategy does not consider the occurrence of resource failure.

A Prioritized user demand algorithm is proposed in [12] that considers user deadline for allocating jobs to different heterogeneous resources from different administrative domains. It produces better makespan and more user satisfaction but data requirement is not considered. While scheduling the jobs, failure rate is not considered. So the scheduled jobs may be failed during execution.

In the existing system, the grid scheduler schedules the jobs to the resources according to the resource response time but the resource failure history is not considered when allocating them. This results in selecting resources that may have higher tendency to fail.

The main objective is to design a fault tolerant scheduling system that schedules the resources and selects the resource which has the lowest tendency to fail. It depends on a new factor called scheduling indicator when selecting the resources. This factor comprises of the response time and the fault rate of grid resources. Whenever a grid scheduler has jobs to schedule on grid resources, it uses the scheduling indicator to generate the scheduling decisions. The scheduling algorithm selects the resources that have the lower response time and the lower fault rate (i.e) resource with minimum scheduling indicator value.

III. MATERIALS AND METHODS

Proposed Methodology

The proposed fault tolerant scheduling algorithm depends on a new factor called scheduling indicator when selecting the resources. This factor comprises of the response time and the fault rate of grid resources. To calculate the fault rate, two parameters Number of failure (N_f) and Number of success (N_s) are used. When a resource fails to complete a job, the value of N_f is incremented by 1. Otherwise, the value of N_s is incremented by 1. The response time is the summation of the job transmission time from the scheduler to the resource on which the job will be executed, the job execution time on that resource and the transmission time of job's execution results from the recourse to the scheduler.

Based on the scheduling indicator value, the scheduler creates a two-dimensional matrix named Scheduling Indicator (SI) matrix. Each entry in the matrix represents the scheduling indicator of each job for each suitable resource in the grid. Finally, each row in the SI matrix is sorted in an ascending order according to the scheduling indicator of each resource. The job is submitted to the resource with minimum scheduling indicator value. Whenever a grid scheduler has jobs to schedule on grid resources, it uses the scheduling indicator value to generate the scheduling decisions.

The proposed scheduling algorithm has maximized the throughput and minimized the makespan of the system. The throughput of the system is the number of jobs executed per unit time and the makespan is the time difference between the start and finish of a sequence of jobs. Job Information includes length of the job, input file size, output file size and bandwidth. Resource information includes its speed, number of success and failures. Based the requirement, 'm' number of resources and 'n' number of jobs are created.The scheduling indicator combines

the response time of the resource and the fault rate of that resource.

Fault rate is manipulated using two parameters N_f (Number of failures) and N_s (Number of successes). N_f is the number of times the resource had failed in executing the job assigned. N_s is the number of times the resource had executed the job successfully. The fault rate P_{fj} calculation is performed using the formula specified in the Equation (1).

$$P_{f_i} = \frac{N_f}{N_s + N_f} \quad (1)$$

Each time a resource fails to complete a job, the value of N_f is increased by 1. Otherwise, the value of N_s is increased by 1. The value of P_{fj} is used by the scheduler when taking scheduling decisions. The most reliable resource will be the resource with the minimum value of P_{fj}.

The response time is the summation of the job transmission time from the scheduler to the resource on which the job will be executed, the job execution time on that resource and the transmission time of job's execution results from the recourse to the scheduler. The response time T_{ij} of a resource j for a job i is defined in the Equation (2).

$$Tij = \tau rj + \tau ej + \tau rr \quad (2)$$

where τrj is the job's transmission time from the scheduler to the resource j, τej is the job's execution time on the resource j and τrr is the time for transferring results from the resource j to the scheduler. τrj can be defined in the Equation (3).

$$T_{rj} = \frac{K_i}{B_j} \quad (3)$$

where K_i is the input file size of the job i and B_j is the bandwidth between the grid scheduler and the resource j on which the job i can be executed. τ_{ej} is be defined in the Equation (4).

$$T_{ej} = \frac{L_i}{RS_j} \quad (4)$$

where L_i is the length of the job i in Million Instructions (MI) and R_{sj} is the speed of the resource j in Million Instructions Per Second (MIPS). The value of τ_{rr} depends on the size of results obtained after executing the job is defined by the Equation (5).

$$T_{rr} = \frac{K_{ir}}{B_j} \quad (5)$$

where K_{ir} is the size of the output file obtained after executing job i.

Based on the scheduling indicator value, the scheduler creates a two-dimensional matrix named SI matrix. Each entry in the matrix represents the scheduling indicator of each job for each suitable resource in the grid. Initially, the scheduler collects all the necessary information about the job such as the length of the jobs, input file size, output file size, bandwidth and the information about the resources such as its speed, number of success and failure. Based on the requirement, the resources and jobs are created with desired characteristics. Then, the values of fault rate, response time and scheduling indicator are calculated. Then, two-dimensional SI matrix is created based on the scheduling indicator values.

Each row in the SI matrix is sorted in an ascending order according to the scheduling indicator of each resource. Finally, the job is submitted to the resource with minimum scheduling indicator value. If the job is completed successfully N_s is increased by 1 otherwise N_f is increased by 1.

IV. RESULTS AND DISCUSSION

Simulation setup

The arrangement of grid resources in GridSim 5.0 and the hierarchy of resources used for evaluating the proposed scheduling algorithm is given in fig.2. Each resource is characterised by number of machines and each machine is characterised by number of processing elements.

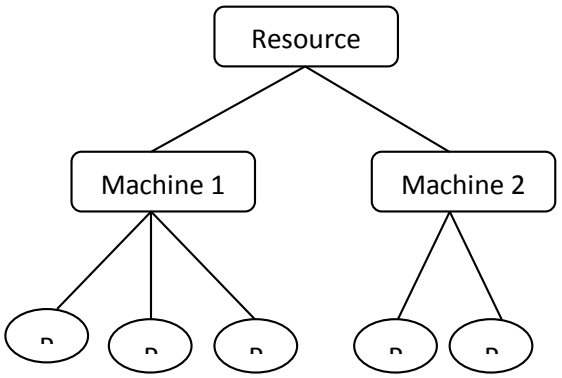

Fig. 2. Arrangement of Grid Resource in Gridsim

Simulation Results

A set of 20, 40, 60, 80 and100 jobs is executed with 10 resources. The makespan of the proposed Fault Tolerant Scheduling (FTS) algorithm is compared between the existing Fault Index Based Scheduling (FIBS) algorithm in the Fig 3. From the figure, it is clear that the makespan of the proposed system is decreased by 15% compared to the existing algorithm.

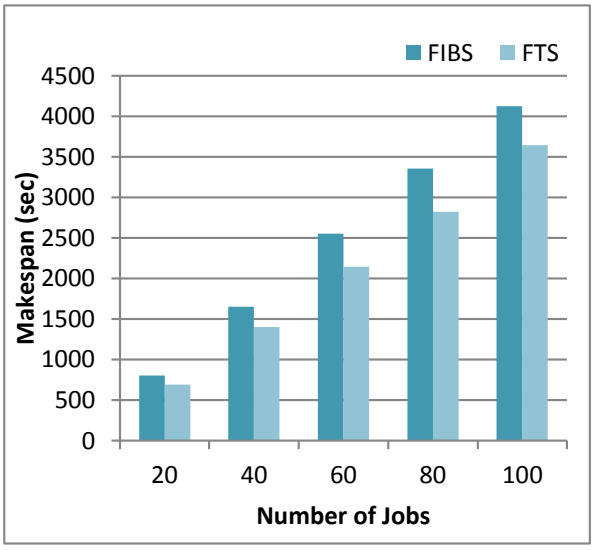

Fig.3 Comparison based on Makespan

V. CONCLUSION

A fault-tolerant scheduling system is proposed that uses the scheduling indicator value when allocating resources to execute the jobs. The throughput of the system is maximized and the makespan is minimized. It is observed that the performance of the proposed system is better than the existing system. If the resource is failed when executing a job, the job is assigned to another resource and it is executed from the beginning. In order to overcome this, checkpoint mechanism can be used. Checkpoint is the ability to save the state of a running job to reduce the fault recovery time. In case of fault, this saved state can be used to resume the execution of the job from the point where the checkpoint was last registered instead of restarting from its beginning. This can reduce the execution time to a large extent.

REFERENCES

I. Lee H, Chung K, Chin S, Lee J, Park S, Yu H (2005), 'A resources management and fault tolerance services in grid computing**',** Journal of Parallel Distributed Computing Vol.65 pp.1305–1317.Mohammed Amoon (2012), 'A fault-tolerant scheduling system for computational grids', Journal of Computers and Electrical Engineering Vol.38 pp.399–412.

II. Sathya SS, Babu K.S (2010), 'Survey of fault tolerant techniques for grid**',** Computer Science Review Vol.4 No.2 pp.101–120.

III. Nandagopal M, Uthariaraj V.R (2010), 'Fault tolerant scheduling strategy for computational grid environment**'**, International Journal of Engineering Science and Technology Vol.2 No.9 pp.4361–4372.

IV. Khan F.G, Qureshi K, Nazir B (2010), 'Performance evolution of fault tolerance techniques in grid computing system**',** Journal of Computers and Electrical Engineering Vol.36 pp.1110–1122.

V. Hwang S. and Kesselman C, A Flexible Framework for Fault Tolerance in the Grid, *Journal of Grid Computing*, Vol.1, 2003, pp. 251-272.

VI. Dabrowski C, Reliability in grid computing, *International Journal of Computer Science*, Vol.8, No.1, 2009, pp. 123-129.

VII. Latchoumy P. and Khader S.A.P, Survey on Fault Tolerance in GridComputing, *International Journal of Computer Science & Engineering Survey (IJCSES)*, Vol.2, No.4, 2011, pp. 97-110.

VIII. Priya B.S, Fault Tolerance and Recovery for Grid Application Reliability using Check Pointing Mechanism, *International Journal of Computer Applications*, Vol.26, No.5, 2011, pp. 32-37.

IX. Suresh P, Balasubramanie P, User Demand Aware Scheduling Algorithm for Data Intensive Tasks in Grid Environment, *European Journal of Scientific Research*, Vol.74, No.4, 2012, pp.609-616.

X. Keerthika P, Kasthuri N, An Efficient Grid Scheduling Algorithm with Fault Tolerance and User Satisfaction, *Mathematical Problems in Engineering*, Volume 2013, Article ID 340294, 2013.

XI. Suresh P, Balasubramanie P and Keerthika P, Prioritized User Demand Approach for Scheduling Meta Tasks on Heterogeneous Grid Environment, *International Journal of Computer Applications*, Volume 23, No.1, 2011.