ISSN: 2277-5528
Impact Factor: 4.015

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & MANAGEMENT

AN ENHANCED FAULT TOLERANT SCHEDULING ALGORITHM FOR GRID ENVIRONMENT

Deepika Pathak[1], Dr. Sharad Gangele[2]
[1]Research Scholar, [2]Prof,
[1,]Department of Computer Science and Application, RKDF University, Bhopal
[2]Department of Computer Science, RKDF Universtiy, Bhopal

ABSTRACT

Lattice processing is getting to be plainly prominent because of the dynamite development in the web, uniting the assets which makes the client to get quick answers for the substantial scale issues. This empowers sharing, determination and total of a wide assortment of geologically disseminated assets. As size of the applications develop, more use of assets for longer timeframes, may prompt expanding number of asset disappointments. At the point when disappointments happen, the execution of the occupations that is appointed to the fizzled assets will be influenced. In this way, adaptation to internal failure is fundamental in such cases. To conquer the disappointment, a planning calculation is suggested that relies on upon another figure called booking marker choosing the assets. This element involves the reaction time and the blame rate of framework assets. The blame rate depends on the achievement and the disappointment of occupation execution. At whatever point a matrix scheduler has employments to plan on framework assets, it utilizes the booking marker to create the planning choices. The asset with least booking pointer esteem gets the occupation.

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 empowers sharing, choice and collection of a wide assortment of geologically disseminated assets. This incorporates supercomputers, stockpiling frameworks, information sources and concentrated gadgets utilized for taking care of substantial scale asset escalated issues in science, building and trade. These issues require an awesome number of PC preparing cycles or the need to handle extensive measure of information. The measure of a lattice may fluctuate from being little, bound to a system of PC workstations inside an enterprise to an overall system.

Computational grids are the solution for all these problems. They offer a helpful approach to associate numerous gadgets (e.g., processors, memory and Input and Output (I/O) - gadgets) so that end clients can consolidate the computational energy of all gadgets for a specific measure of time. For instance, if a client needs to make some CPU expending estimations, the client could incidentally obtain CPU-time from a matrix with a much lower fetched than get the time from a super PC. A grid could be created in all environments where end users have a computer with memory and CPU. Information network gives a foundation to bolster information stockpiling, information revelation, information dealing with, information distribution and information control of vast volume of information really put away in different heterogeneous databases and record systems[2]. It manages the controlled sharing and administration of a lot of dispersed information.

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 develop, more utilization of assets for longer timeframes may prompt increment in number of asset disappointments. At the point when disappointments happen, the execution of the occupations doled out to the fizzled assets will be influenced. Along these lines, a blame tolerant administration is essential in matrix condition. Adaptation to internal failure is the capacity to save the conveyance of expected administrations regardless of disappointments inside the network 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 employments put together by the client are executed by the computational matrix by allotting them to the assets with Quality of Service (QoS) necessities. Fig.1 demonstrates the essential lattice booking model.

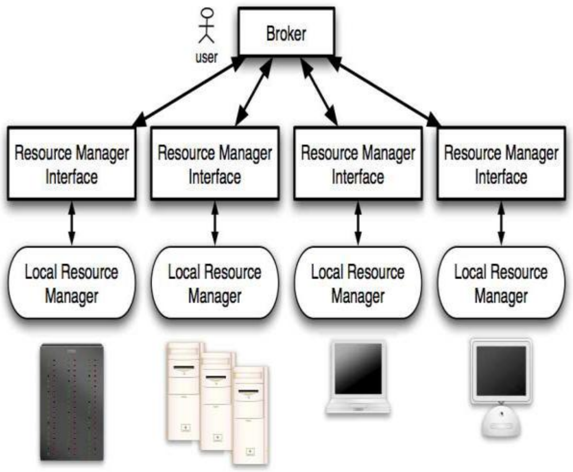
user
Broker
Resource Manager Interface
Resource Manager Interface
Resource Manager Interface
Resource Manager Interface
Local Resource Manager
Local Resource Manager
Local Resource Manager
Local Resource Manager

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.

Blame tolerant components are expected to shroud the event of deficiencies, or the sudden inaccessibility of assets. In spite of the fact that booking and adaptation to non-critical failure have been customarily considered freely from each other, there is a solid relationship between's them. Actually, each time an adaptation to internal failure activity must be performed.

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 issues by consolidating the checkpoint replication with Minimum Time To Release (MTTR) work planning calculation and blame list is tended to by [4]. Time to discharge incorporates the administration time of the occupation, holding up time in the line, exchange time of info and yield information to and from the asset. MTTR calculation limits an opportunity to discharge by choosing a computational asset in light of employment necessities, work attributes and equipment components of the assets. 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.

A versatile blame tolerant employment booking technique for economy based lattices is proposed in [5]. The proposed procedure keeps up a blame record of lattice assets. The blame tolerant calendar administrator keeps up blame history about matrix assets and updates Fault Index (FI) of a network asset by getting demands from the merchant. It powerfully refreshes the blame file in view of fruitful or unsuccessful finish of an appointed undertaking. At whatever point a framework asset merchant has undertakings to plan on network assets, it makes utilization of the blame list from the blame tolerant calendar supervisor notwithstanding utilizing a period enhancement heuristic. FI is not an appropriate pointer to speak to the asset disappointment history as it can't be decremented underneath a specific utmost. FI of a lattice asset is increased each time the asset does not finish the doled out occupation and is decremented at whatever point the asset finishes the appointed employment effectively. On the off chance that the blame file is zero, at that point the asset with the base reaction time is chosen paying little respect to its disappointment history. This outcomes in choosing assets that may tend to fall flat.

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 assets are demonstrated in view of the framework unwavering quality. Dependability of a lattice figuring asset is measured by mean time to disappointment (MTTF), the normal time that the matrix asset works without disappointment. Mean time to repair (MTTR) is the normal time it takes to repair the Grid figuring asset after disappointment. The MTTR measures the downtime of the registering asset.

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 system is proposed in [9] to accomplish adaptation to internal failure. The check pointing process occasionally spares the condition of a procedure running on a registering asset so that, in case of asset disappointment, it can continue on an alternate asset. In the event that any asset disappointment happens, it summons the vital imitations so as to meet the client application unwavering quality necessities.

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 current framework, the network scheduler plans the occupations to the assets as per the asset reaction time however the asset disappointment history is not considered while distributing them. This outcomes in choosing assets that may tend to come up short. The primary target is to outline a blame tolerant planning framework that timetables the assets and chooses the asset which has the most reduced propensity to fall flat. It relies on upon another element called planning marker while choosing the assets.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.

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.

In light of the planning pointer esteem, the scheduler makes a two-dimensional framework named Scheduling Indicator (SI) network. Every section in the network speaks to the booking marker of each occupation for each appropriate asset in the framework. At long last, each line in the SI network is sorted in a rising request as indicated by the booking pointer of every asset. The employment is submitted to the asset with least planning pointer esteem. At whatever point a network scheduler has employments to plan on framework assets, it utilizes the planning marker incentive to produce the booking choices.

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).

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

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).

$$Tij = \mathsf{T}rj + \mathsf{T}ej + \mathsf{T}rr \qquad (2)$$

where $\mathsf{T}rj$ is the job's transmission time from the scheduler to the resource j, $\mathsf{T}ej$is the job's execution time on the resource j and $\mathsf{T}rr$ is the time for transferring results from the resource j to the scheduler. $\mathsf{T}rj$can be defined in the Equation (3).

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

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. T_{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 T_{rr}depends on the size of results obtained after executing the job is defined by the Equation (5).

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

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

In view of the planning marker esteem, the scheduler makes a two-dimensional network named SI lattice. Every section in the network speaks to the planning pointer of each occupation for each reasonable asset in the matrix. At first, the scheduler gathers all the vital data about the employment, for example, the length of the occupations, input document estimate, yield record size, transmission capacity and the data about the assets, for example, its speed, number of achievement and disappointment. In view of the necessity, the assets and employments are made with craved qualities. At that point, the estimations of blame rate, reaction time and booking marker are figured. At that point, two-dimensional SI lattice is made in light of the booking marker esteems. 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.

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.

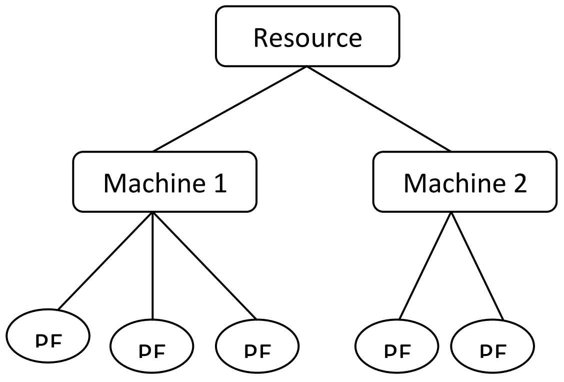

Fig. 2. Arrangement of Grid Resource in Gridsim

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.

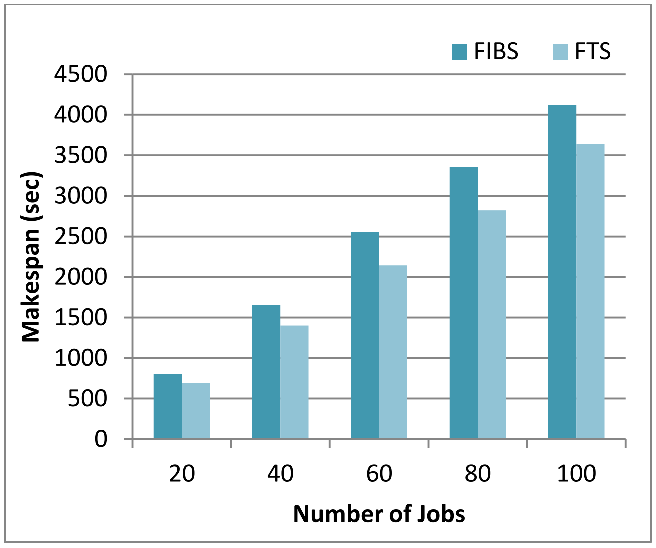

Fig.3 Comparison based on Makespan

CONCLUSION

A blame tolerant booking framework is suggested that uses the planning pointer esteem while allotting assets to execute the occupations. The throughput of the framework is augmented and the makespan is limited. It is watched that the execution of the proposed framework is superior to the current framework. In the event that the asset is fizzled when executing an occupation, the employment is alloted to another asset and it is executed from the earliest starting point. With a specific end goal to conquer this, checkpoint instrument can be utilized. Checkpoint is the capacity to spare the condition of a running occupation to diminish the blame recuperation time. If there should be an occurrence of blame, this spared state can be utilized to continue the execution of the employment from the point where the checkpoint was last enlisted as opposed to restarting from its start. This can decrease the execution time to a vast degree.

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.