ISSN: 2277-5528
Impact Factor: 4.015

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & MANAGEMENT

AN IMPROVED 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, Dr Dr. APJ Abdul Kalam University, Indore
[2]Department of Computer Science, RKDF Universtiy, Bhopal

ABSTRACT

Framework processing is getting to be noticeably famous because of the spectacular 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 topographically conveyed assets. As size of the applications develop, more utilization of assets for longer timeframes, may prompt expanding number of asset disappointments. At the point when disappointments happen, the execution of the employments that is allotted to the fizzled assets will be influenced. In this way, adaptation to internal failure is vital in such cases. To beat the disappointment, a booking calculation is suggested that relies upon another factor called planning pointer in choosing the assets. This factor contains the reaction time and the blame rate of network assets. The blame rate depends on the achievement and the disappointment of occupation execution. At whatever point a framework scheduler has employments to plan on network assets, it utilizes the booking pointer to create the planning choices. The asset with least planning pointer esteem gets the occupation.

INTRODUCTION

In the present unavoidable world, the touchy matrix registering conditions have turned out to be noteworthy that they are frequently alluded to be the world's single and most capable PC arrangements. Beforehand, the assets were accessible worldwide in each framework. Because of the enormous development of Internet and approach of lattice figuring, the assets are united to make the client, get quick answers for their employments. Matrix figuring is characterized as the controlled and composed asset sharing and critical thinking in powerful, multi - institutional virtual associations. It includes the real systems administration administrations and associations of a conceivably boundless number of processing gadgets inside a network. Lattice figuring makes progress toward a perfect Central Processing Unit (CPU) cycles and capacity of a large number of frameworks crosswise over overall users[1]. This empowers sharing, determination and conglomeration of a wide assortment of topographically conveyed assets. This incorporates supercomputers, stockpiling frameworks, information sources and concentrated gadgets utilized for taking care of substantial scale asset serious issues in science, designing and business. These issues require an incredible number of PC preparing cycles or the need to process expansive measure of information. The span of a lattice may shift from being little, bound to a system of PC workstations inside an organization to an overall system. Computational lattices are the answer for every one of these issues. They offer an advantageous approach to interface numerous gadgets (e.g., processors, memory and Input and Output (I/O) - gadgets) so 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 devouring figurings, the client could at times acquire CPU-time from a framework with a much lower cost than obtain the time from a super PC. A matrix could be made in all situations where end clients have a PC with memory and CPU. Information lattice gives a foundation to help information stockpiling, information disclosure, information dealing with, information production and information control of substantial volume of information really put away in different heterogeneous databases and document 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 planning framework must consider the booking of occupations including the mapping of "n" employments to "m" assets. Planning is finished by utilizing programming called work scheduler. Multifaceted nature of matrices starts from solid varieties in the lattice accessibility and an expansion in the likelihood of assets to come up short than customary parallel and dispersed frameworks.

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 allocated to the

fizzled assets will be influenced. In this way, a blame tolerant administration is imperative in network condition. Adaptation to internal failure is the capacity to save the conveyance of expected administrations regardless of disappointments inside the matrix itself. Shortcomings happen when a framework asset can't finish the doled out occupation. Deficiencies happen because of asset disappointment, work disappointment or system disappointment. The asset disappointment is considered in this work. The employments put together by the client are executed by the computational framework by assigning them to the assets with Quality of Service (QoS) necessities. Fig.1 demonstrates the fundamental lattice planning model.

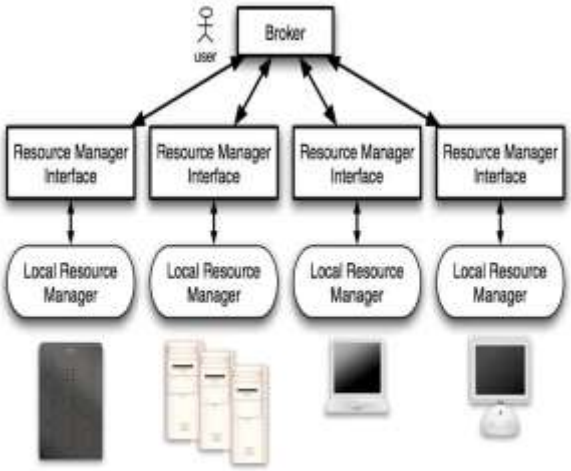

Fig.1 Grid Scheduling Model

An incorporated intermediary is the single point for the entire foundation and oversees specifically the asset administrator interfaces that collaborate straightforwardly with the neighborhood asset supervisors. Every one of the clients present the undertakings to the brought together broker.Each asset contrasts from different assets by numerous ways that incorporates number of preparing components, handling speed, inward planning strategy and its heap factor and so on. Thus each employment varies from different occupations by execution time, due date, time zone and so forth. Blame tolerant systems are expected to shroud the event of shortcomings, or the sudden inaccessibility of assets. Despite the fact that planning and adaptation to internal failure have been customarily considered autonomously from each other, there is a solid connection between's them. Truly, each time an adaptation to non-critical failure activity must be performed.

RELATED WORKS

Blame tolerant planning is an essential issue for computational network frameworks, as matrices ordinarily comprise of unequivocally fluctuating and geologically appropriated assets. 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 information and yield information to and from the asset. MTTR calculation limits an opportunity to discharge by choosing a computational asset in view of employment prerequisites, work qualities and equipment components of the assets. When settling on planning choices, the scheduler utilizes the blame file and the reaction time of assets. It sets the employment checkpoints in view of the asset disappointment rate. A basic perspective for a programmed recuperation is the accessibility of checkpoint records. A methodology to build the accessibility 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 blame tolerant calendar supervisor keeps up blame history about matrix assets and updates Fault Index (FI) of a lattice asset by getting demands from the agent. It progressively refreshes the blame file in view of fruitful or unsuccessful consummation of an allocated assignment. At whatever point a matrix asset dealer has assignments to plan on framework assets, it makes utilization of the blame record from the blame tolerant calendar administrator notwithstanding utilizing a period enhancement 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) instrument and an adaptable disappointment dealing with system is proposed in [6]. The FDS empowers the discovery of both errand accidents and client characterized special cases. The Grid-WFS is based over FDS, which enables clients to accomplish disappointment recuperation in an assortment of routes relying upon the prerequisites and limitations of their applications. The assets are displayed in light of the framework dependability. Dependability of a network figuring asset is measured by mean time to disappointment (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.

Different blame recuperation instruments, for example, checkpointing, replication and rescheduling are talked about in [7]. Taking checkpoints is the procedure of occasionally sparing the condition of a running procedure to tough stockpiling. This permits a procedure that neglects to be restarted from the point its state was last spared, or its checkpoint on an alternate asset. Replication: Replication implies keeping up an adequate number of imitations, or duplicates, of a procedure executing in parallel on various assets so that no less than one reproduction 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 intermittently spares the condition of a procedure running on a figuring asset so that, in case of asset disappointment, it can continue on an alternate asset. On the off chance that any asset disappointment happens, it summons the essential copies keeping in mind the end goal to meet the client application unwavering quality requirements.In our past work [10], we have proposed a productive blame tolerant planning calculation (FTMM) which depends on information exchange time and disappointment rate. Framework execution is likewise accomplished by decreasing the sit without moving time of the assets and conveying the unmapped undertakings similarly among the accessible assets. A planning system that considers client due date and correspondence time for information serious undertakings with diminished makespan, high hit rate and lessened correspondence overhead is presented in [11]. This methodology does not consider the event of asset disappointment. 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.

MATERIALS AND METHODS

Proposed Methodology

The proposed blame tolerant planning calculation relies upon another factor called booking marker while choosing the assets. This factor contains the reaction time and the blame rate of lattice assets. To compute the blame rate, two parameters Number of disappointment (Nf) and Number of achievement (Ns) are utilized. At the point when an asset neglects to finish a vocation, the estimation of Nf is increased by 1. Something else, the estimation of Ns is augmented by 1. The reaction time is the summation of the occupation transmission time from the scheduler to the

asset on which the employment will be executed, the occupation execution time on that asset and the transmission time of employment's execution comes about because of the response 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 employment is submitted to the asset with least planning marker esteem. At whatever point a lattice scheduler has occupations to plan on network assets, it utilizes the booking marker incentive to create the planning choices.

The proposed booking calculation has augmented the throughput and limited the makespan of the system.The throughput of the framework is the quantity of occupations executed per unit time and the makespan is the time distinction between the begin and complete of a succession of employments. Employment Information incorporates length of the occupation, input document measure, yield record size and data transfer capacity. Asset data incorporates its speed, number of progress and disappointments. Based the prerequisite, "m" number of assets and "n" number of occupations are created.The booking pointer consolidates the reaction time of the asset and the blame rate of that asset. Blame rate is controlled utilizing two parameters Nf (Number of disappointments) and Ns (Number of victories). Nf is the quantity of times the asset had bombed in executing the employment doled out. Ns is the quantity of times the asset had executed the employment effectively. The blame rate Pfj computation is performed utilizing the recipe indicated 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 Nf is increased by 1. Otherwise, the value of Ns is increased by 1. The value of Pfj is used by the scheduler when taking scheduling decisions. The most reliable resource will be the resource with the minimum value of Pfj. 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 Tij of a resource j for a job i is defined in the Equation (2).

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

whereτrj is the job's transmission time from the scheduler to the resource j, $\tau ej$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. τrjcan 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. τ_{ej} is be defined in the Equation (4).

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

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).

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

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

In light of the booking pointer esteem, the scheduler makes a two-dimensional network named SI framework. Every section in the framework speaks to the booking pointer of each occupation for each appropriate asset in the network. At first, the scheduler gathers all the important data about the employment, for example, the length of the occupations, input record measure, yield document size, transmission capacity and the data about the assets, for example, its speed, number of accomplishment and disappointment. In light of the necessity, the assets and occupations are made with wanted qualities. At that point, the estimations of blame rate, reaction time and planning pointer are ascertained. At that point, two-dimensional SI network is made in light of the booking pointer 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 course of action of network assets in GridSim 5.0 and the progressive system of assets utilized for assessing the proposed booking calculation is given in fig.2. Every asset is portrayed by number of machines and each machine is described by number of handling components.

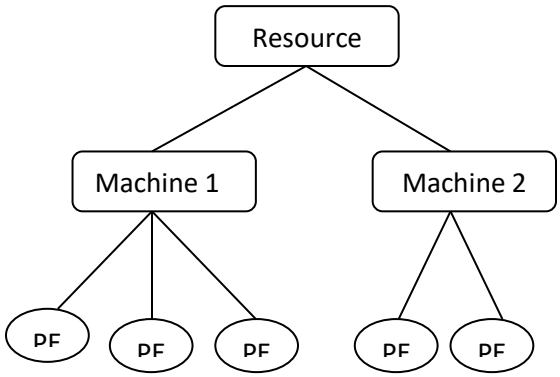

Fig. 2. Arrangement of Grid Resource in Gridsim

Simulation Results

An arrangement of 20, 40, 60, 80 and100 employments is executed with 10 assets. The makespan of the proposed Fault Tolerant Scheduling (FTS) calculation is thought about between the current Fault Index Based Scheduling (FIBS) calculation in the Fig 3. From the figure, unmistakably the makespan of the proposed framework is diminished by 15% contrasted with the current calculation.

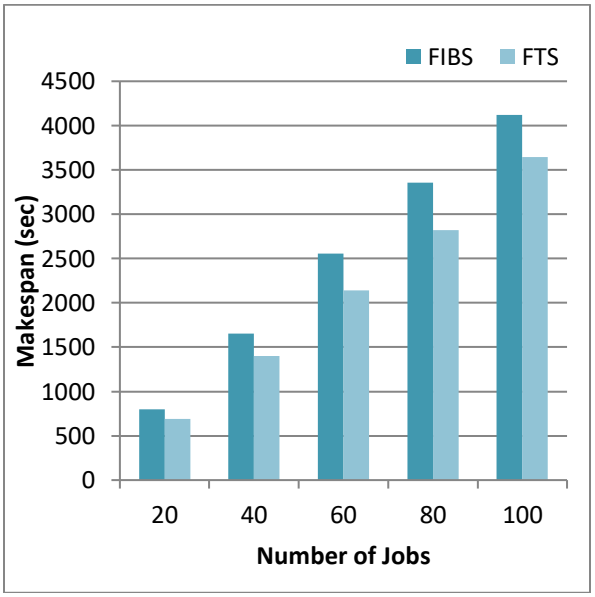

Fig.3 Comparison based on Makespan

CONCLUSION

A blame tolerant planning framework is recommended that uses the booking marker esteem while apportioning assets to execute the occupations. The throughput of the framework is expanded 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 work, the occupation is allocated to another asset and it is executed from the earliest starting point. So as 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 arise an occurrence of blame, this spared state can be utilized to continue the execution of the occupation from the point where the checkpoint was last enrolled as opposed to restarting from its start. This can diminish the execution time to a substantial 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.