Fault Tolerance to Minimize Makespan during Job Scheduling in Grid Computing

Amandeep Kaur
Student Masters of Technology
Department of Computer Science and Engineering
Sri guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India

Abstract— In Grid Computing there are network of computers connected to each other. These computers are distributed through different networks and belong to different administrative domains. There are most of the challenging demands on the network and job Scheduling is one of them. Makespan time influences the job scheduling and itself depends upon fault tolerance of resources. This research work focus upon detecting the faults and then apply certain actions to recover the faults. This paper concentrates on two types of faults that are omission and intermittent faults. Replication technique is applied in this process to fault tolerance. Jobs are scheduled on the resources in Grid with great care. Makespan reduction is also one of the objectives of the work so that efficient job scheduling will achieve in Grid Computing.

Key words: Grid Computing, Job Scheduling, Makespan, Fault Tolerance, K-means clustering, Replication

I. INTRODUCTION

Grid computing provides the consistent, dependable and transparent access to the distributed resources. These resources are shared by multiple organizations to provide services for applications. Grid system is like a distributed computing such as “super virtual computer”. Super virtual computers consist of loosely coupled computers. All the systems work together to finish the same task. But in grid computing parallel computing can be done on the system having CPU, power supply, storage and network interface. Grid is further classified into different types that is:

- Computational grids: This replaces the need of super computers and helps the processing of computational tasks.
- Collaboration grid: the demand for best hardware resources is fulfilled by this collaboration grid.
- Utility Grid: sharing of CPU, softwares and other peripherals is done under this utility grid.
- Network grid: in grid there should be best network for better communication among distributed resources.
- Data grid: this grid system is used for the storage purpose like data discovery, handling, publication, etc.

Scheduling is one of the best part of grid computing. Scheduling is to reduce the completion time of any application by properly allocating it to the processors. Scheduling is further divided into two subparts that is matching and mapping. In matching it find out the resource matching the requirements of the task and in mapping it allocate the task to the corresponding resource.

Job schedulers are ment to manage the jobs like partitioning the jobs into subtasks, allocation of resources to the tasks etc. these are divided into three types that is meta schedulers which are the local schedulers, Meta scheduler forwards the jobs to local and cluster schedulers.

II. RELATED WORK

- Vaishali K. Patel and Prof. Mitula H. Pandya [1] improved the greedy algorithm by using genetic and base bound algorithm. Their target was to select maximum number of compatible activities that can be done by a single machine. Two types of techniques exists to solve this problem that is Greedy and Dynamic Programming Method. They concentrate upon the algorithm that gives maximum number of activities and minimum makespan time.
Inderpreet Kaur and Sarpeet Singh [2] focus on to test the crash and omission transient failure in resource scheduling. They presented an overview of fault tolerance and its techniques, task replication and most fitting resource allocation algorithm. Two type of faults that is crash and omission faults were detected and using replication task assigned to next node.

P. Keerthika and N. Kasthuri [3] proposed BSA algorithm considered proactive fault tolerance and user deadline while scheduling the jobs. BSA algorithm considered the user deadline of each task and the failure rate of each resource at the time of scheduling. The performance of the proposed algorithm was evaluated using Grid Sim based on makespan and number of jobs that were completed successfully within user deadline. The proposed BSA algorithm showed reeducation in makespan and better hit rate with higher user satisfaction and fault tolerance.

Dipti Sharma and Mr. Pradeep Mittal [4] presented an algorithm which helps in scheduling computational resource to the jobs in efficient way. In this paper they proposed a scheduling algorithm EDSRTF and compare it with two other algorithms FCFS and LJF in terms of waiting and turnaround time of jobs. In EDSRTF algorithm scheduling is done according to the deadline of jobs and their remaining time. In comparison to other scheduling algorithms Average waiting time and average turnaround time of proposed algorithm is less. Which proves that among them EDSRTF is best is best algorithm.

Ritu Garg and Awadshesh Kumar Singh [7] introduced some of the faults, errors and the ways to achieve fault tolerance. The major techniques used in any fault-tolerant grid management system are Replication, Check pointing and job migration. Response time of any task is reduced by replication. Task level and workflow level is used for the dependent task grid.

D.Maruthanayagam and Dr. R.Uma Rani [10] focuses on applying one of the rapidly growing non-deterministic optimization algorithms, the ant colony algorithm in Grid computing. They work upon the ACO by combining the Ant Colony Algorithm with the RASA( resource Acquisition Scheduling Algorithm). Initially they find the problem resources and then after that total execution time is equal to the makespan of the solution and then try to attempt to move or swap set of jobs from the processor having some problem to another resource that has the minimum and maximum of makespan as compare with all the other resources.

III. RESEARCH METHODOLOGY

Whole methodology of the research work is given below:

- There will be one master server on which ACO (Ant Colony Optimization) algorithm is applied. This algorithm helps to check all the available resources in grid system either they are working properly or not. Also check the number of processes running on them. Then master server will send socket message to every resource and all the resources reply back their processed data back to main server which it will store in its database and send the required reply to its corresponding client.

- Users are unaware about the background functioning of the grid system. For them it seems to be like their whole work is done by the main server but instead main server is further linked to various other resources. If users want to send any request they send it to main server.

- The main server then make cluster of multiple request in ascending order based on Process duration and then it make dynamic array list of all process in queue. Those processes whose processing times are close to each other come under one cluster and likewise various clusters are formed. Then main server will forward the whole cluster to its matching resource where all the processes will be processed acc to the ascending order of their execution time.

- The master server will check the resource with least no of processes and send the process to the resource which is having least no of processes running on it. If due to certain conditions there is any failure in this resource then we will choose the next resource having least resources in ascending order. Main server will also check omission and intermittent failure factors while sending the user request to the resource. Check the efficiency of all the resources that is which server finishes its work with expected or less turnaround (response) time. Only then we can forward next job to it. If it did not performed well in past, there is no question to assign new task to it. Then we will choose other resources for this purpose. If any server is overloaded that is performing work more than its efficiency then there are chances of its crash and we can shift its job to another server. While checking, if due to certain condition (power off etc) server does not respond or it might be crashed then omission failure generates and to remove this we can shift its work to another server. To keep us safe from lost our data we use replication concept here. After the completion of the tasks every resource will replicate its updated data on all the resources in grid so that due to any condition if this resource will fail we can have or continue our work from other resources which are having the same updated data. Omission failure reduction process checks the connection only at the beginning but what if connection fails in between when only some of the files were replicated or transferred? To overcome this problem we are using a thread which will check the intermittent failures also after every neno second. By this we can send the rest of the files which are left remain due to connection failure after establishing connection again. This intermittent failure is not only checked between the main server and other resources instead this is checked among all the resources while replicating data from one resource to another.

- If there is any failure in the resource then popup will show.

- If resource does not reply back to main server then it means there is any problem on the resource. So we will check the failure factors in the form of their true or false status. If it shows true then it means their corresponding failure generates and if false it means it is not.
After completion process by resources then they update to other resource for make consistency, so that if unfortunately this resource damage then we can get its belonging data from another resources. So we do it to save our data for future use.

Each server will update to each other they track like how many files get update on one connection so that uncopied files will copied later for make consistency.

The bar graph shown below represents the makespan time of both the algorithms.

Figure 3 shows the makespan time of BSA and proposed algorithm for case 1.

![Fig. 3: Graph of makespan time in case 1](image_url)

Figure 4 shows the makespan time of BSA and proposed algorithm for case 2.

![Fig. 4: Graph of makespan time in case 2](image_url)

Figure 5 shows the makespan time of BSA and proposed algorithm for case 3.

![Fig. 5: Graph of makespan time in case 3](image_url)

### IV. RESULTS

Here we are having table of makespan time of BSA and proposed algorithm.

<table>
<thead>
<tr>
<th>S.no.</th>
<th>BSA (in sec.)</th>
<th>Proposed algorithm (in sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>1284011</td>
<td>218190</td>
</tr>
<tr>
<td>Case 2</td>
<td>564011</td>
<td>438565</td>
</tr>
<tr>
<td>Case 3</td>
<td>1594319</td>
<td>481077</td>
</tr>
</tbody>
</table>

Table 1:
All the above plots show that in all the cases makespan time of Proposed algorithm is less as compare to the makespan time of BSA algorithm.

V. CONCLUSION
This thesis work used the ACO algorithm and k-means clustering for scheduling the jobs on the systems and check for any fault occur. This paper works on two types of faults that are omission fault and intermittent fault. If any of these faults generated first detected then proposed algorithm follow appropriate action to recover it along with following replication technique. After that results of makespan time of proposed algorithm with that of BSA algorithm are compared using bar graph in the Net Beans Java IDE.

Results show that proposed scheme helps to achieve fault tolerance and reduce the makespan time.

REFERENCES