Fair Scheduling and Optimal Fault Tolerance Approaches to Increase the Performance of Grid Environment

D.G. Thakare\textsuperscript{1} and P.B. Niranjane\textsuperscript{2}

\textsuperscript{1}II Year ME Student, Department of CSE, Babasaheb Naik College of Engineering, Pusad, (MS)
\textsuperscript{2}Assistant Professor Department of CSE, Babasaheb Naik College of Engineering, Pusad, (MS)

Abstract—Grid computing is became an important technology in distributed computing technology. The Concept is focused on grid computing has Load balancing, Fault tolerance and recovery from Fault failure. Grid computing is a set for techniques and methods applied for the coordinated use of multiple servers. These servers are specialized and works as a single, logic integrated system. Grid computing is defined as a technology that allows strengthening, accessing and managing IT resources in a distributed computing environment. Network Security addresses a wide range of issues, such as: authentication, data integrity, access control and updates. Grid system and technologies, in order to secure a place on the corporations market and their use in important IT departments. Fault tolerance concept—Binding the developers to predict and deal with all the faults in a system leads to a significant increase of the application complexity. For this reason, grid computing would be provided with mechanisms for fault tolerant computing, applications from various partners. In this paper we study the Load balancing and Fault tolerance. Load balancing is the technique that maintain the workload on sites. To achieve high output, and resource utilization we propose a Fair scheduling algorithm and an optimal fault tolerances that need to proposed the system and they conducted using Grid Simulation Toolkit (GridSim).

Keywords—Distributed computing; Fault tolerance; Grid computing; Optimal Fault Tolerance’s GridSim; Load balancing; Scheduling

I. INTRODUCTION

Grid is became an emerging technology in the IT infrastructure. In grid we can easily share hardware, data, software and resources. The concept behind grid computing is more important that we can’t able to buy more resources but we can borrow computational resources from where it’s available. An important characteristic of grid computing is revolved around resource sharing. The resources are shared among various applications. In this case load balancing plays an vital role. So there is a need for a fair scheduling approach. Scheduling is an important process to optimize the load in the system. The grid environment needs a proper scheduling and load balancing algorithms to increase the overall performance of the system. Grid computing is a another form of distributed computing that includes coordination and sharing computing, application, data and storage or network resources across dynamic and geographically dispersed organization. Grid technologies promise to change the way organizations tackle complex computational problems. The main concept in fault model is “Resource reclaiming” that invoked when the primary site finished the job before the estimated time. The backup slot are removed and assigned for the new job which avoids the backup overloading. It having two types of load balancing policy in grid environment: Static load balancing policy and Dynamic load balancing policy. In grid if the job failed due to some faults in the sites it should be tolerated. At the primary site, a backup is made for each job. If the primary site is failed the backup always succeeds. There are two types of load balancing policy in grid environment: Static load balancing policy and Dynamic load balancing policy. The static load balancing policy is not in useful to grid environment because the load may vary with respect to time. But in dynamic load balancing policy the workstations are constantly monitored. The selection of the policy is done at run time and also uses the current load information for decision making. Dynamic Load balancing
policy will not require the priority task information to allocate/reallocate the resource. Dynamic load balancing algorithm will give better performance than static load balancing algorithm.

II. RELATED WORK

In grid computing there are numerous job scheduling algorithm are used to utilize the resources effectively. This process will increase the execution performance and balance the system load. This will happen when new resource join or old resource exit from the grid environment because grid is a dynamic environment the resources in the environment will change over time so designing a job scheduling algorithm to dynamically change according to the variation in resource requires numerous and complicated factor[1]

This will makes when either new resource join or old resource exit from the grid environment because grid is a dynamic environment and resources in the environment makes change over time so designing a job scheduling algorithm to dynamically change according to the variation in resource requires numerous and complicated factors. It will manage the load across the nodes in fairly manner. We focused it on that the difference between “heaviest loaded” node and “lightest loaded” node need to be minimized[2].

Previous work has been done to facilitate real-time computing in heterogeneous systems. Huh et al. proposed a solution for the dynamic resource management problem in real-time heterogeneous systems. A probabilistic model for a client/server heterogeneous multimedia system was presented in [2]. These algorithms, however, also could not tolerate any permanent processor failures. While eFRD tolerates any one processor’s permanent failure, the algorithm presented in [1], also a real-time scheduling algorithm for tasks with precedence constraint, does not support fault-tolerance. eFRD schedules the backup copy to start after its primary copy’s scheduled execution time, thus avoiding unnecessary execution of the backup copy if the primary copy completes successfully. Dima et al. also devised an off-line real-time and fault-tolerant scheduling algorithm to handle both processor and communication link failures [3]. However, this algorithm must execute the backup copy of a task simultaneously with its primary copy. Tasks considered in eFRD can either be confined by

Fig. 1. Process Flow of the Grid Scheduler
precedence constraints or be independent, and eFRD may be generalized to consider heterogeneous systems, where homogeneity is just a special case.

Fault-tolerance must be considered in the design of scheduling algorithms, because occurrences of faults are often unpredictable in computer systems. Previous work has been done to facilitate real-time computing in heterogeneous systems. Huh et al. proposed a solution for the dynamic resource management problem in real-time heterogeneous systems. GCS have become important as building blocks for fault-tolerant distributed systems. Such services enable processors located in a fault-prone network to operate collectively as a group, using the services to multicast messages to group. Our distributed problem has an analogous counterpart in the shared-memory model of computation, called the collect problem[4]

III. SYSTEM MODEL

In this project model, every site consists of one each machine and each machine consists of one or more than one processors. we have studied it from designed diagram 1, it provided with global and local grid scheduler which is a software component that present within each site. The system must execute the following operations:-
1. If the local/remote site has to connect in to the grid performance, then resource information services like computation speed, storage capacity, bandwidth communication, etc., need to be massage to the Grid Information Server.
2. Local Grid Schedular manager incise the job with its first request to the Local balancing decision maker that select the minimum loaded site.
3. The grid scheduler admin the state of sites that execute the job requests
4. Site selector administrate the loaded site to select on the job dispatcher
5. Future decision making hold on the coordination of the resource to the last request with the timestamp
6. Then user got the job to be assembled with time stamp with GIS to the Fault manager assign it

IV. FLOW CHART

![Flow Chart Image]
A. Designed Approach

At start the grid client will submit the job to the grid scheduler. Then local grid scheduler manager balanced the job with minimum loaded site that having candidate set of nearest request. also the load balancing decision maker manage the load with its various sake of interest in site selector to select the site by local executions. If the computing nodes get failed due to some resource failure then the fault detector will send the failure message to the fault manage

B. Determination on the Candidate Sites

The grid environment is dynamic so each site in the grid environment has more chances to enter in to the busy state
1. Tasks are aperiodic, i.e., task arrivals are unknown a priori. Every task Ti has the attributes arrival time (ai), ready time (ri), worst case computation time (ci) and a hard deadline (di).
2. The versions of a task have identical attributes and resource requirements. The backup version typically undo the effect of primary’s failed So site selecting is done by not only considering the[5]

Interaction between a Grid service and a client happens in a request-reply fashion using strictly-defined interfaces and a certain encoding of data

C. Fault tolerant

The backup is scheduled for each primary site. Grid service notifications are a natural mechanism for solving all three of these problems because state updates and failures are inherently asynchronous events. Fault-tolerance is an essential requirement for real-time systems faults. In this paper, we investigate an efficient off-line scheduling algorithm generating schedules. Workload parameters are chosen in such a way that they are either based on those used in the literature or represent reasonably realistic workloads.[6]

V. FAIR SCHEDULING APPROACH

To achieve high throughput and high resource utilization Fair scheduling approach is proposed. Our aim is to assign the job to the fair site which completed the task with minimum response time. The simulation model consists of a collection of computers connected by a communication network. Jobs arriving at the system are distributed to the computers. In this section, we investigate the effectiveness of load balancing schemes by varying the speed skewness.

\[ \text{OPj involves minimizing a convex function over a convex feasible region and the first-order Kuhn–Tucker reconditions are necessary and sufficient for optimality [7].} \]

Let \( 0, i, 0, i = 1, \ldots, n \) denote the Lagrange multipliers [7].

The Lagrangian is

\[ L(sj_1, \ldots, sj_n, 1, \ldots, n) = n \sum_{i=1}^{n} sj_i - sj_i - 1 - i = n \sum_{i=1}^{n} sj_i. (7) \]

The Kuhn–Tucker conditions imply that \( sj_i, i = 1, \ldots, n \)

is the optimal solution to OPj if and only if there exists \( 0, i, 0, i = 1, \ldots, n \) such that

\( Lsj_i = 0, \)
\( L = 0, \)
\( isji = 0, i \neq 0, sj_i = 0, i = 1, \ldots, n. \)
These conditions become
\( \begin{align*}
  ji \\
  (ji - sjij)^2 - i = 0, i = 1, \ldots, n, \\
  n i=1 \\
  sj_i = 1, \\
  isji = 0, i \neq 0, sj_i = 0, i = 1, \ldots, n. \\
  These are equivalent to \\
  =ji \\
  (ji - sjij)^2, if sj_i > 0, 1 \ i \ n, \\
  ji \\
  (ji - sjij)^2, if sj_i = 0, 1 \ i \ n, \\
  n i=1 \\
  sj_i = 1, sj_i = 0, i = 1, \ldots, n
\end{align*} \)

**TABLE 2: JOB ARRIVAL SITUATION**

<table>
<thead>
<tr>
<th>User</th>
<th>1</th>
<th>2</th>
<th>3–6</th>
<th>7</th>
<th>8–10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_j )</td>
<td>0.3</td>
<td>0.2</td>
<td>0.1</td>
<td>0.07</td>
<td>0.01</td>
</tr>
</tbody>
</table>

**VI. OPTIMAL FAULT TOLERANCES APPROACH**

Assume there are 4 jobs with the following resource requests 5, 9, 15, 21. As per the proposed approach the all the jobs must be given equal importance irrespective of external or internal request.[9]

The maximum consumable resource for each job needs to be maintained. The approach divides the total processor capacity 50 among the 4 job requests as \( 50/4 = 12.5 \) units of resource. The first and second job request is less then 12.5 units (below the average demand rate) so the job will receive the requested resource. The third and fourth job’s demand rate is more than 12.5 units (above the average demand rate). So in the first iteration of the approach, third and forth job is assigned with the average demand rate (i.e., 12.5 Units of resource each). The residue of first iteration is \( 50-39(5+9+12.5+12.5) = 11 \) units.[10]

At the end of iteration the residue will be calculated. The approach is repeated until the

**TABLE 3. SIMULATED RESULTS OF THE FAIR SCHEDULING APPROACH**

<table>
<thead>
<tr>
<th>Job Request No.</th>
<th>Job Demand Rates</th>
<th>Fair Scheduling (First Iteration)</th>
<th>Fair Scheduling (Second Iteration)</th>
<th>Fair Rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>12.5</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>12.5</td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>Residue</td>
<td>50-39=11</td>
<td>50-47=3</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
algorithm breaks down the procedure of GA into two parts and hence the crossover and mutation procedure is performed by slave nodes selected by master node[11]

VII. CONCLUSION

In this paper, an approach for load balancing and scheduling is proposed. The Throughput and Performance of the grid environment greatly improve by an optimal load balancing approach. Here a fair scheduling approach with equal opportunity to all the jobs is designed. The fair scheduling approach follows the hybrid scheduling by calculating the residue value for each job for a number of iterations until the residue gets down to zero. This approach is linear and iterative in nature which eliminates the fluctuations in the response time.

REFERENCES


