Performance Improvement of Grid System Using Alchemy Middleware

Patel Ochchhav S.

Lecturer, LDRP-ITR Engineering College, Gandhinagar
Kadi Serva Vishwa Vidyalaya University, Gandhinagar

Abstract—Grid system is made by software, hardware and middleware such as alchemy grid middleware. Grid system in which load balancing and scheduling algorithm attempts to improve the performance of the grid system. In grid system Scheduling and Load Balancing is closely related to each other for which load balancing and scheduling algorithms attempt to improve the performance of the grid system. Ideal grid system requires efficient Scheduling and Load balancing mechanisms in a network. This paper defines how the performance of the grid system increases using alchemy middleware with practical implementation. There are three types of failure in a grid system where manager node may be failed, executor node may be failed or both manager and executor nodes failed. In alchemy, Manager is responsible for entire control of the grid system such as Scheduling as well as Load Balancing. So we have optimized the Load balancing execution time of a grid [3].

Keywords—Grid middleware, Load balancing, Alchemy, Peer to Peer, Cluster computing, Internet computing, Grid Area routing protocol.

I. INTRODUCTION

Grid computing is utilizing all the resources of the grid network for solving a large problem such as scientific problem as well as real time application. So grid computing is a combination of all the resources from multiple administrative domains to achieve a specific task. [1] In alchemy grid middleware there are four parts such as alchemy manager, alchemy executor, owner and CrossPlatform manager. One large problem should be divided into smaller task and send to the executor nodes for solving the small task. After completed all small task from the executors whose results are send to the manager node. Here manager should be decided which task is allocated to the suitable executor node. During execution time any executor node will be failed then manager node should be dividing the work of failure node to the working node in a network. This paper discussed about implementation where used the ten executor nodes and one manager node in a grid network and execute the two applications one is prime number generator and second is pi calculator where how the load balancing and scheduling is done efficiently. There are three types of failure possibilities, such as manager node might be failed, executor node might be failed and both manager and executor nodes might be failed. There are three technologies are used behind the grid system one is cluster computing, second is peer to peer computing and third is internet computing.

II. WHY MIDDLEWARE USED IN A GRID SYSTEM?

In today’s world 80% computer working based on windows platform so it would like to create the grid network based on windows platform. A grid is created by installing software services, or middleware on a set of networked computers where, the middleware provides facilities such as Location of hardware and software resources which are lies in a grid environment, distributed scheduling of resources and tasks as well as user authentication. There are two protocol used in grid system such as GFTP and GARP. Where GFTP is a same as the File transfer protocol (FTP) is responsible for sending and receiving the Information from one node to another node in a grid.
environment and GARP is the grid area routing protocol which is responsible for maintain the state information of the grid nodes like response time, waiting time, job execution time etc…,

III. ALCHEMY MIDDLEWARE

In Alchemy, the owner node generates application and Creates threads also thread may be generated by manager node at run time. It submits threads to the manager Node and the manager node store them in a queue. The manager node distributes threads to the Executor nodes in a network based on first in first out priority. The executor node executes it and sends result back to the manager node. When all threads are finished, the manager node combines them and sends final result to the owner node.

A. MANAGER

The Manager manages the execution of grid applications and provides services associated with thread execution. The Executors connected with the Manager which in turn keeps track of their availability. Threads received from the Owner are placed in a pool and scheduled to be executed on the various available Executors. A priority for each thread can be explicitly specified when it is created within the Owner, but is assigned the highest priority by default if none is specified. Threads are scheduled on a Priority and First Come First Served (FCFS) basis, in that order. The Executors return completed threads to the Manager node. [16]

B. EXECUTOR

The Executor got the threads from the Manager and executes them. An Executor can be configured to be dedicated, meaning the resource is centrally managed by the Manager, or non-dedicated (peer to peer), meaning that the resource is managed on a volunteer based by the user. For non-dedicated execution, there is one-way communication between the Executor and the Manager. In this case, the resource that the Executor resides on is managed on a volunteer basis since it requests threads to execute from the Manager. Where two-way communication is possible and dedicated execution is desired the Executor exposes an interface so that the Manager may communicate with it directly. In this case, the Manager explicitly instructs the Executor to execute threads, resulting in centralized management of the resource where the Executor resides. Thus, Alchemy’s execution model provides the dual benefit of communication.

C. OWNER

Grid applications created using the Alchemi Application Programming Interface (API) are executed on the Owner component. The Owner provides an interface with respect to grid applications between the application developer and the grid. Hence it “owns” the application and provides services
associated with the ownership of an application and its constituent threads. The Owner submits threads to the Manager and collects completed threads on behalf of the application developer via the Alchemi API.

D. CROSSPLATFORM MANAGER
Cross platform manager should be used for platform independent jobs execution. This is an optional sub component of the Manager, is a generic web services interface that exposes a portion of the functionality of the Manager in order to enable Alchemi to manage the execution of platform independent grid jobs. Jobs submitted to the Cross-Platform Manager are translated into a form that is accepted by the Manager (i.e. grid threads), which are then scheduled and executed as normal in the fashion described above. Thus, in addition to supporting the grid-enabling of existing applications, the Cross-Platform Manager enables other grid middleware to interoperate with and leverage Alchemi on any platform that supports web services.

IV. SCHEDULING AND LOAD BALANCING IN A GRID SYSTEM
A. SCHEDULING:
The job scheduling system is responsible to select best suitable machines in a grid environment for user jobs.

Types of job scheduling infrastructure [3]
- Centralized scheduling.
- Decentralized scheduling.
- Hierarchical scheduling.

TYPES OF JOB SCHEDULING ALGORITHM
There are three types of job scheduling algorithm.
1) First come first serve.
2) Random.
3) Backfill.

FIRST COME FIRST SERVE: The scheduler starts the jobs in the order of their submission. If not enough resources are currently available, the scheduler waits until the job can be started.

RANDOM: In a randomize algorithm the next job will be selected randomly from all the submitted jobs. This method is not suitable for efficient scheduling.

BACKFILL: This type of algorithm is special FCFS algorithms that prevent unnecessary idle time caused by wide jobs. [3]

B. LOAD BALANCING
The main objective of a load balancing consists primarily to optimize the average response time of Applications.

THREE FORM OF LOAD BALANCING:
LOAD SHARING
This is the coarsest form of load distribution. Load may only be placed on idle resources, and can be viewed as a binary problem.
LOAD BALANCING
Load balancing is the finest. Load balancing attempts to ensure that the workload on each resource is within a small degree, or balance criterion, of the workload present on every other resource in the system.

LOAD LEVELLING
To describe the middle ground between the two extremes of load sharing and load balancing. In a grid network Rather than trying to obtain a strictly even distribution of load across all resources, or simply utilizing idle resources. Load leveling seeks to avoid congestion on any resource. Load balancing system can be generalized into four basic steps. Such as Monitoring resource load and state, Exchanging load and state information between resources, calculating the new load distribution, Updating data movement.

V. PROBLEM DISCUSSION AND RELATED SOLUTION
We know that there are three possibilities of failure the execution in a grid middleware. In first possibilities the manager should transmit the number of task to the suitable executor nodes. The execution of executor node has finished the execution and sends that result to the manager node. After completion of the all the task on all executor node results will be transfer to the manager node. But it should possible to failure the manager node as well as executor nodes in a network. All the information of nodes regarding execution are stored in a manager node as well as control of the grid network by alchemy manager node so may be fail the entire grid network. [3]

In second possibilities executor node may be failed in a network so it should require the balance the load of that failure node to the working nodes in a network. Load balancing of failure node is balanced by manager node to the survive the work to the other working node in a centralized grid system. But in a decentralized grid system load should be balanced to other grid working node without requirement of communication with centralized node.

In third possibilities both manager and executor node will be failed than entire grid network will fall down. So it should require the store the execution backup to other system with time interval. In alchemy’s default situation, if any of the executor nodes fails to complete its assigned job then this task is rescheduled to another executor and the task is executed again from its initial state.

Generally, if any job associated with a grid application unable to complete during its runtime in the executor then the scheduling algorithm just marks the job as failed and reset but it does not keep any track of the percentage of the work that has already been done. From the implementation result, we have found this technique to improve the performance of the grid system.

![Figure 2: Load balancing with one failure node](image-url)
A. IMPLEMENTATION RESULT

<table>
<thead>
<tr>
<th>No. of Executors</th>
<th>CPU power In GHz</th>
<th>Input for Prime number generator</th>
<th>Requ. time for exec before applied strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>7.26</td>
<td>1000000</td>
<td>1.70750</td>
</tr>
<tr>
<td>3</td>
<td>7.26</td>
<td>2000000</td>
<td>2.585625</td>
</tr>
<tr>
<td>3</td>
<td>7.26</td>
<td>200000000</td>
<td>8.989375</td>
</tr>
<tr>
<td>5</td>
<td>10.198</td>
<td>2000000</td>
<td>1.90000</td>
</tr>
<tr>
<td>5</td>
<td>10.198</td>
<td>200000000</td>
<td>2.9706</td>
</tr>
<tr>
<td>5</td>
<td>10.198</td>
<td>2000000000</td>
<td>12.4375</td>
</tr>
<tr>
<td>10</td>
<td>27.826</td>
<td>2000000000</td>
<td>4.99812</td>
</tr>
<tr>
<td>12</td>
<td>31.289</td>
<td>2000000000</td>
<td>4.90895</td>
</tr>
</tbody>
</table>

Figure 3: Output of grid application with default strategy of alchemy.

<table>
<thead>
<tr>
<th>No. of Executors</th>
<th>CPU power In GHz</th>
<th>Input for Prime number generator</th>
<th>Requ. time for exec before applied strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>7.26</td>
<td>1000000</td>
<td>1.55810</td>
</tr>
<tr>
<td>3</td>
<td>7.26</td>
<td>2000000000</td>
<td>2.45568</td>
</tr>
<tr>
<td>3</td>
<td>7.26</td>
<td>2000000000</td>
<td>8.10236</td>
</tr>
<tr>
<td>5</td>
<td>10.198</td>
<td>2000000000</td>
<td>1.60001</td>
</tr>
<tr>
<td>5</td>
<td>10.198</td>
<td>20000000000</td>
<td>2.39401</td>
</tr>
<tr>
<td>5</td>
<td>10.198</td>
<td>20000000000</td>
<td>11.3785</td>
</tr>
<tr>
<td>10</td>
<td>27.826</td>
<td>20000000000</td>
<td>3.99856</td>
</tr>
<tr>
<td>12</td>
<td>31.289</td>
<td>200000000000</td>
<td>3.05243</td>
</tr>
</tbody>
</table>

Figure 4: Output of grid application with proposed strategy.

Figure 5: Load balancing without proposed strategy

Figure 6: Load balancing with proposed strategy
Here grid application has been executed using various number of executors where number of executors 3 and processing capacity has been 7.26GHz and input parameter for prime number generator was 1000000 digits and found the randomize factor from that input value. We have stopped the one executor node using manager node because all the control of the network done by the grid manager. Using the proposed strategy for the alchemy we have got the less execution time than after use of the default strategy. Also we have been experiment with more processor and got the conclusion can see in figure 4, figure 5 and figure 6.

VI. CONCLUSION

This paper describes the performance of the grid system could be increased using of some input parameter as well as changes default strategy. It is necessary to improve the performance of the grid system to optimize the Load balancing time as well as Scheduling in a grid network. Also find the failure to repair rate of each grid node using reliability model and programming method. So it should possible using the proposed technique to optimize the load balancing execution time so improve the performance of the grid system also make a grid system more reliable.

VII. FUTURE WORK

In this implementation, Load Balancing done on only through Alchemy Manager but we have no here use the more than one manager in a group of grid network and Load will be shared based on Intra working nodes as well as inter working nodes in a grid network.

REFERENCES


WEB REFERENCES