A New Algorithm for Load Balancing and Fault Tolerance mechanism in Computational Grid

Dr. Ramesh T. Prajapati¹, Dr. Dushyantsinh Rathod², Mr. Tejas Kadiya³

Department of Computer Engineering
¹Indrashil Institute of Science and Technology, Rajpur, Kadi, Gujarat
²Aditya Silver Oak Institute of Technology, Ahmadabad, Gujarat, India
³Indrashil Institute of Science and Technology, Rajpur, Kadi, Gujarat
{ramesh.9998590139@gmail.com}

ABSTRACT:
Grid computing is sharing and coordinating resources to multiple users. All resources are stored in different location any user can access that resources from different location. Parallel execution of application computation cannot complete if nodes failures so the automatic handling of faults necessary for grid environment Load Balancing is a process of distributed workload evenly in all nodes to get better resource utilization, minimum time and avoid overload. The proposed work presents dynamic load balancing and fault tolerant technique for improve the performance of grid computing. For this I had setup a computational lattice in light of the Alchemi middleware for implementation of fault tolerance and load balancing. We found that fault tolerance and dynamic load balancing are suitable for grid users which aim to execute their application speedily and failure of central node back up manager will take control and balance load of all processor to avoid grid failure.

Keywords: Grid computing, Alchemi, Fault Tolerance, Load balancing, Resources.

INTRODUCTION
In past several years the demand for computing power in computational grid environments is continuously increasing. The popularity of the Internet and the availability of powerful and high-speed computing resources and network technologies with optimal and low-cost changes the way of computing. The primary inspiration for developing circulated frameworks is to get substantial scale asset sharing at moderate cost. To get this objective, the interconnection between broadly conveyed heterogeneous assets has turned into a need for building conveyed frameworks[1].Progresses in systems administration innovation and computational foundation make it conceivable to build substantial scale elite appropriated registering situations that give reliable, predictable and inescapable access to top of the line calculation. Matrix processing has been broadly observed as a stage past the routine conveyed registering, consolidating unavoidable high-data transmission, rapid processing, wise sensors and substantial scale database into a consistent pool of oversaw and facilitated assets. Be that as it may, the significant test in such very heterogeneous and complex registering condition is to outline a productive blame tolerant framework. The vast majorities of the prior proposed blame tolerant framework were created for neighborhood and are not productive with regards to wide territory conveyed frameworks and are described by an abnormal state of asynchrony, long message deferral and high likelihood of message misfortune. Grid Computing defined as applying resources from thousands or many computers in a network to a single problem, usually one that requires to access large amounts of data. Network [14] is sort of parallel and circulated framework that empowers the sharing, determination and collection of geologically conveyed assets progressively at run time contingent upon their accessibility, ability, and execution cost and client nature of administrations. Load adjusting is a system to upgrade assets, using parallelism, abusing throughput ad lib, and to slice reaction time through a proper dissemination of the applications [9].Work movement is the main proficient approach to ensure that submitted employments are finished dependably and productively if there should be an occurrence of process disappointment, processor disappointment, hub crash, arrange disappointment, framework execution corruption, correspondence delay; expansion of new machines progressively despite the fact that the asset disappointment happens which changes the dispersed environment [11].

These are critical issues in Load Balancing: A surprising pinnacle can be steered to generally sit out of gear machines in the Grid. On the off chance that the Grid is as of now completely used, the least need job being completed on the matrix can be incidentally postpone or even scratched off and carry out further next to prepare space for the big need job. As an essential administration for Grids is of essential significance in such frameworks. Computational Grid is a collection of distributed, possibly heterogeneous resources which can be used as an ensemble to execute large-scale applications. Computational Grid is also called metacomputer. Term computational Grid comes from an analogy with the electric power Grid [11].

LITERATURE REVIEW
Due to unavailability of network or development difficulty or faulty resources, fault may occur in the results or performance may be degraded. There are many levels of adaptation to internal failure, the least being the capacity to proceed with operation in case of a power disappointment. Many blame tolerant PC frameworks reflect all operations - that is, each operation is performed on at least two copy frameworks, so on the off chance that one falls flat the other can assume control [5].
The fault tolerance is "to preserve the delivery of expected services despite the presence of fault-caused errors within the system itself. Errors are detected and corrected, and permanent faults are located and removed while the system continues to deliver acceptable service." From a user's point of view, a distributed application should continue despite failures. The fault tolerance [4] has become the main topic of research. Till now there is no single system that can be called as the complete system that will handle all the faults in grids. Grid is a dynamic system and the nodes can join and leave voluntarily. Because of computational Grid heterogeneity, scale and complexity, faults become likely. Therefore, Grid infrastructure must have mechanisms to deal with faults while also providing efficient and reliable services to its end. There are several types of faults occur in Grid computing as below [11].

1. Hardware faults:
   Due to faulty hardware components such as CPU, memory, and storage devices which arise the hardware faults. Due to faulty processors arise CPU faults in incorrect output. Memory faults are occurring due to damaged memory in the RAM, ROM or cache.

2. Application and Operating System Faults:
   Due to some specific faults occur in Application and operating system failures. The application gets large amount of data and it is not releases, if the is memory leakage. Deadlock, inefficient or improper resource management in Operating system faults. Resource unavailable: The application fails to execute when resource are not available.

3. Network Faults:
   The resources are connected with different types of distributed networks In Grid computing [11]. Node failure: By operating system error, network error or physical error, it may go down the node. Packet loss: The packets are loss when links are crashed. Corrupted packet: The transformation of Packets from one end to another, it can be corrupted.

4. Processor faults:
   Machine or operating system crashes.

5. Media faults:
   Disk head crashes.

6. Service expiry fault:
   The service time of a resource may expire while application is using the resources in grid.

Load adjusting is a procedure to upgrade assets, using parallelism, misusing throughput act of spontaneity, and to slice reaction time through a suitable appropriation of the application [8].

In the decentralized load adjusting calculation proposed in [23] for a Grid domain. In spite of the fact that this work endeavors to incorporate the correspondence inerterness between two hubs amid the activating procedure on their model, it didn't consider the genuine cost for an occupation exchange. In [24, 25], a sender processor gathers status data about neighboring processors by imparting each heap adjusting minute. This can prompt regular message moves that outcomes in huge correspondence overhead which is unwanted. Preemptive and non-preemptive process relocation methods are proposed in [22]. In this procedure relocation happens proficiently by considering memory use and load on processor.

The gaining popularity of high speed emerge the problem of load balancing. A load is the number of jobs in the waiting queue and can be light and heavy according to their work. Load balancing is a process of improving the performance of computational grid system in such a way that all the computing node involve in the grid utilized equally as much as possible. Load balancing is an important function of grid system to distribute the workload among available computing nodes to improve minimizes execution times and maximizes node utilization and overall system performance.

Dimple & Atul [20] proposed an ant based framework to balance the load. In their work the author proposed an active ant at client side and the server side both. The client ant is responsible for the request whereas the server ant is responsible for replying the request. The authors improved the server performance in their work.

Sandeep et al., [16] presented the performance analysis of static and dynamic load balancing algorithms. Comparison is done the various parameters of overload rejection, fault tolerance, accuracy and stability etc. Selection of load balancing algorithm is based on situation in which work load is assigned i.e. at run time or compile time. Static algorithms are proved to be more stable as compare to dynamic load balancing algorithms.

Several load indices have been proposed in the literature, like CPU queue length, average CPU queue length, CPU utilization, etc. The success of a load balancing algorithm depends from stability of the number of messages (small overhead), support environment, low cost update of the workload, and short mean response time which is a significant measurement for a user [8]. It is also essential to measure the communication cost induced by a load balancing operation.
Gurveer Kaur Brar and Amit Chhabra (2016) present an extensive review of different meta-heuristic techniques that are applicable in grid and cloud systems to generate optimal load balancing solution. Mushu Li, Peter He, and Lian Zhao (2016) present an exact approach in order to allocate the elastic loads based on the inelastic load’s information considering the group and node power upper constraints. For online approach, the reference level is computed dynamically using historical demand data to minimize the fluctuation in the grid, and the elastic loads can only be scheduled in the future time slots.

PROBLEM STATEMENT

Our primary concentration is on the improvement of adaptation to internal failure framework for computational networks [7]. For this we had setup a computational grid in view of the Alchemi middleware. In the event of disappointment of the focal administrator, reinforcement director will take its control and dodges the grid to come up short. In grid registering the assets are shared and dynamic in nature, which influences application execution. If Failure of the Central Manager [9] than the System is fail: There is no backup in case of manager failure. So, the manager is stop, connected Executors will fail and thread as well. The running thread “hangs” on the grid. So, the running threads are not answering but the heart beating threads area live. Until it is restarted, the Executor remains hanged. This is not a suitable solution. Sometimes executors are in connected state even they are disconnected.

Nature of resources is dynamic and shared which affects the performance of application. There are two important functions, Management of resource and Workload which provided at the usage level of the matrix. The main objective of load adjusting is to examine problems due to which load balancing technique is required in grid computing and also to transfer work load from heavy loaded nodes to lightly loaded nodes based on migration policy. This research evaluates the existing Load Balancing algorithm and find out the performance of grid. There are five policies for implementation of Load Balancing algorithms [6]. The usage of Information Policy, in current heap adjust calculation utilizes intermittent approach which is tedious. The Triggering approach utilizes Queue length which isn’t chosen how to stack adjusting process is finished. For execution of calculation utilizing determination approach in view of Job length which can be utilized to settle on choice about choice of assignment for movement.

PROPOSED ALGORITHM

A. Utilizing Alchemi Based Computational Grids Alchemi.NET [12] is a free source programming system that Permits you to effortlessly total the processing force of Arranged system into a large computer and create applications to keep running on the Grid. Alchemi.NET incorporates:

A. Algorithm for Fault Tolerance:

1. Start
2. User submit the application to the manager
3. a. Manager distributes the application to number of executor
   Which are connected to manager. Every 2 second executor
   Nodes check the status of manager is alive. (Check the
   Status of Manager using IP address and Port No) Generate
   The replication of Database.
   b. Continue the work between executor and manager and generate the checkpoint.
4. If work delay is greater than 2 second then
   a. Restart the failed thread on another backup manager node.
5. Return the computed final result to the Manager.
6. Manager sends the result to the user
7. Stop
In proposed algorithm we implemented fault tolerance system for manager node which centrally assigns work to all connected node. at that time after failure of manager node all executor node directly connected to another backup manager node and completed execution. We got maximum response time to execute the application. A .NET application programming and interface and devices to create .NET grid applications and network empowered applications while working with the setup network, the disappointment of the concentrated director causes the entire framework to hangs down. Till the administrator doesn’t restart, the network stays difficult to reach. So to evade such sort of network disappointment, we have presented the idea of the reinforcement administrator. The reinforcement administrator additionally utilizes the heart beat marvel to question the status of supervisor. The means for actualizing the reinforcement supervisor are: After each pulse interim, the pioneer hub sends a bundle to the reinforcement director.

B.Algorithm for Load Balancing with fault tolerance

1. Start
2. Start the Manager node and connect all executor nodes to Manager.
3. Get the availability of all resources Manager assigns Application to all executor nodes whose workload is Lightly loaded.
4. Initialize the Queue Length.
5. Monitor the system at regular interval and check the Executor nodes are lightly or heavily loaded node.
6. Heavily loaded node found then we find lightly loaded Node.
7. Lightly loaded node found then we transfer heavily loaded Node to lightly loaded node.
8. When task is completed and there are no more requests From next node then stop
In this proposed algorithm for load balancing we found how many nodes have more load and light node. Based on heavy node we transfer load to light node to continue execute of application.

V. RESULT DISCUSSION

Following results are display for fault tolerance

![Figure 3 Alchemi Manager and Executor Connect](image)

Figure 3 Alchemi Manager and Executor Connect

Figure: 3 After Alchemi.NET executor Start and wait for Connection Above images shows that the Alchemi.NET Manager and Executor have started and they are further waiting for the heartbeat signal for establishing the connection with each other for further processing.

Creating the Port Scanner We need port scanner kind of application to check for the status of the manager. In our port scanner application, application tends to scan port 9000 of the remote system. Where manager is running at regular interval of time. The code for this application is written in c#. The two main classes used for this are: ‘Socket’ and ‘IPEndPoint’. User has to provide the IP address of the system where manager is running. In case when the manager fails due to any reason, a message showing manager failure is displayed at Backup Manager. Now the backup manager starts acting as the new central manager

![Figure 4 Automatic Backup Manager User Interface](image)

Figure 4 Automatic Backup Manager User Interface

![Figure 5 Databases from PC-1](image)

Figure 5 Databases from PC-1
Figure 6 Automatic Backup done successfully from PC-2

Figure 7 Port Scan User Interface

Figure 8 Manager Port not working User Interface

Figure 9 Connect Backup Manager User Interface
In above figures we implemented automatic backup of one computer to another computer. They executed their tasks with less time. Here Execution of Prime Number Generator Application with failure of first manager and connect with second manager. Following results are display for load balancing with fault tolerance.
In above figures we implemented for load balancing with heavily loaded and lightly loaded node. Here we are finding which executor nodes have heavy loaded and lightly loaded node. Accordingly we adjust load and execute the application with execution time.

Comparison of existing and proposed fault tolerance and load balancing algorithm.
Table 1 Comparison of existing Fault tolerance and Proposed Fault tolerance

<table>
<thead>
<tr>
<th>Prime of 10000000</th>
<th>Input Range to find out prime number</th>
<th>Total No of Executor Running</th>
<th>Total Time to Complete execution (Proposed Method)</th>
<th>Total Time to Complete execution (Existing Method)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>06.4896</td>
<td>08.0939</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>07.0189</td>
<td>09.8875</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>09.0110</td>
<td>11.9864</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>11.6578</td>
<td>14.5436</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>13.5860</td>
<td>16.8123</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>15.8435</td>
<td>19.4337</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>20.4432</td>
<td>26.8420</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Comparison of existing load balancing and Proposed Dynamic Load balancing algorithm

In previous comparison table 1 and table 2 we found our proposed algorithm is better than existing algorithm in terms of time and response time very less.

**VI. CONCLUSIONS AND FUTURE ENHANCEMENTS**

Grid works on various tasks within a network, but it is also capable of working on specialized applications. It is designed to solve problems that are too big for a supercomputer while maintaining the flexibility to process numerous smaller problems. Grid is dynamic and heterogeneous nature. So resource management is very important task for grid. We proposed new fault tolerance and dynamic load balancing algorithm. Manager node allocated some task to different nodes at that time the manager node fail so another backup manager node will come and all executor nodes continue their work with minimum time and response time. Another issue load balancing means balancing the load from heavily loaded node to lightly loaded node so we can utilize idle nodes so we got minimum response time to complete execution of particular application. Based on above implementation we have executed prime no application with minimum time and minimum response time. We will further extend this work in big data and cloud computing

**REFERENCES**


[12] Raissa Medeiros, Walfredo Cirne, Francisco Brasilheiro, Jacques Sauvé, “Faults in Grids: Why are they so bad and what can be done about it?” GRIDS’03


