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

Dr. Ramesh T. Prajapati[1], Dr. Dushyantsinh Rathod[2], Mr. Tejas Kadiya[3]

Department of Computer Engineering
[1]Indrashil Institute of Science and Technology, Rajpur, Kadi, Gujarat
[2]Aditya Silver Oak Institute of Technology, Ahmadabad, Gujarat, India
[3]Indrashil 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 an 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 inertness 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) presents 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
```

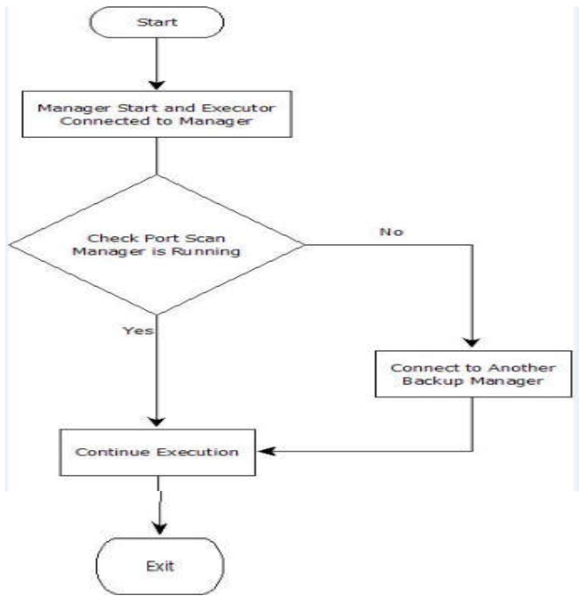

Start
Manager Start and Executor Connected to Manager
Check Port Scan Manager is Running
No
Yes
Connect to Another Backup Manager
Continue Execution
Exit

Figure 1 Proposed Architecture of Fault Tolerance

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.

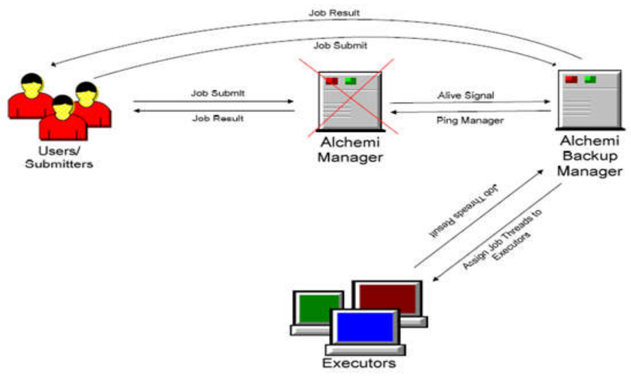


Figure 2 Grid Backup Manager Take the control while Failure of Manager [5]

Figure 2 demonstrates the new association built up between the agents and the reinforcement director. We attempt to utilize these means for giving the support to reinforcement administrator for Alchemi based computational lattices. From that point the reinforcement administrator gets to the database to control the framework.

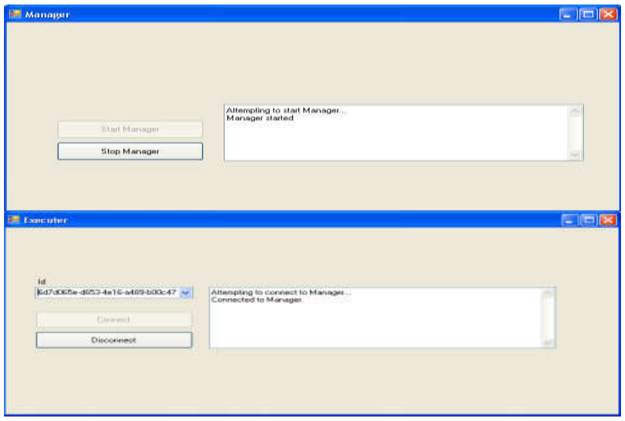
B.Algorithn for Load Balancing with fault tolerance

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



```
Attempting to start Manager...
Manager started
```

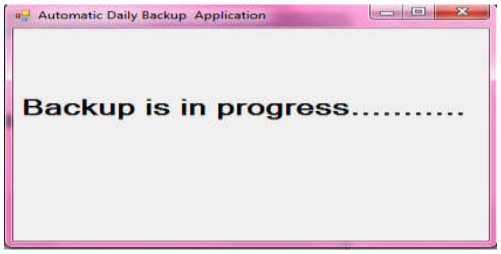
```
6d7d065e-d653-4e16-a489-b00c47
```

```
Attempting to connect to Manager...
Connected to Manager.
```

Figure 3 Alchemi Manager and Executor Connect

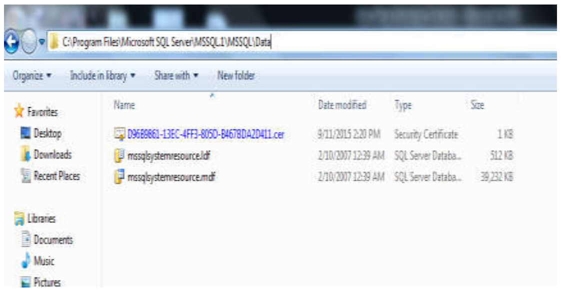
Figure: 3 After Alchemi.NET executor Start and wait for Connection Above images shows that the Alchemi.NET Manager and Executor have started and they are further waiting for the heartbeat signal for establishing the connection with each other for further processing.

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



Backup is in progress...........

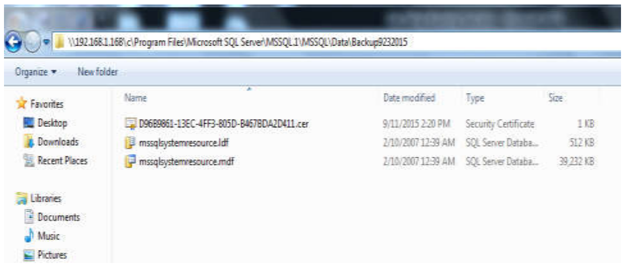
Figure 4 Automatic Backup Manager User Interface



C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data

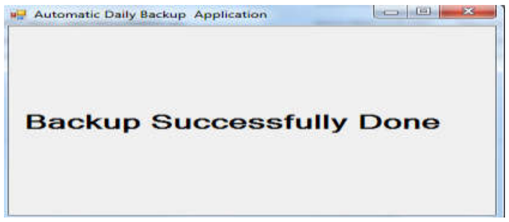
Name	Date modified	Type	Size
D96B9861-13EC-4FF3-805D-B467BDA2D411.cer	9/11/2015 2:20 PM	Security Certificate	1 KB
mssqlsystemresource.ldf	2/10/2007 12:39 AM	SQL Server Databa...	512 KB
mssqlsystemresource.mdf	2/10/2007 12:39 AM	SQL Server Databa...	39,232 KB

Figure 5 Databases from PC-1



\\192.168.1.168\c\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\Backup9232015

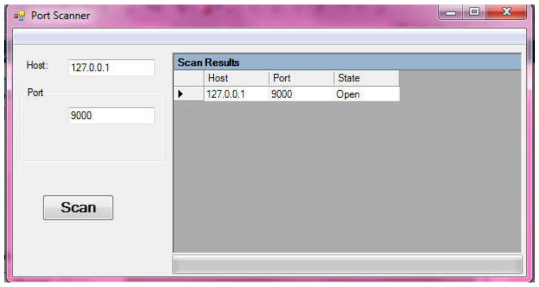
Name	Date modified	Type	Size
D96B9861-13EC-4FF3-805D-B467BDA2D411.cer	9/11/2015 2:20 PM	Security Certificate	1 KB
mssqlsystemresource.ldf	2/10/2007 12:39 AM	SQL Server Databa...	512 KB
mssqlsystemresource.mdf	2/10/2007 12:39 AM	SQL Server Databa...	39,232 KB



Automatic Daily Backup Application

Backup Successfully Done

Figure 6 Automatic Backup done successfully from PC-2



Port Scanner

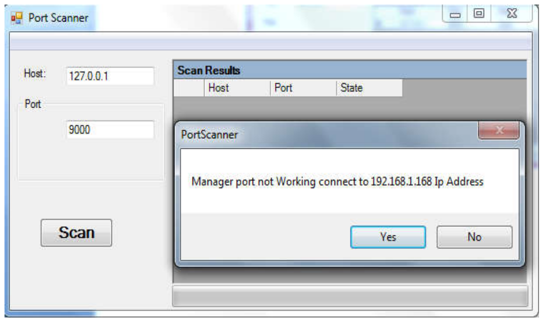
Host: 127.0.0.1

Port 9000

Scan Results

	Host	Port	State
▶	127.0.0.1	9000	Open

Figure 7 Port Scan User Interface



Port Scanner

Host: 127.0.0.1

Port 9000

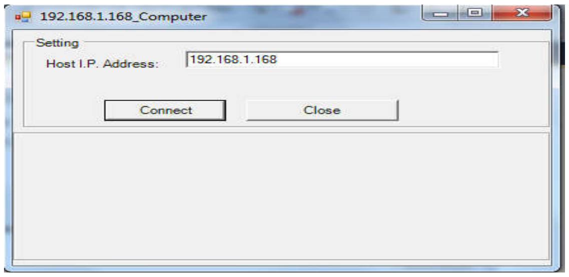
Scan Results

Host	Port	State

PortScanner

Manager port not Working connect to 192.168.1.168 Ip Address

Figure 8 Manager Port not working User Interface

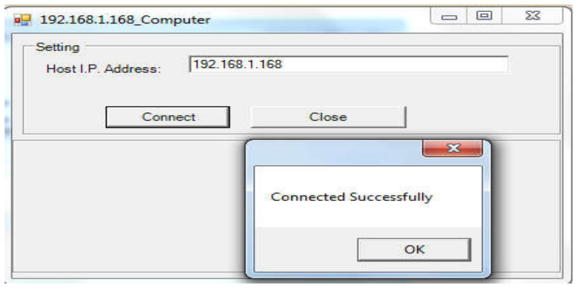


192.168.1.168_Computer

Setting

Host I.P. Address: 192.168.1.168

Figure 9 Connect Backup Manager User Interface



```
192.168.1.168_Computer
Setting
Host I.P. Address: 192.168.1.168
Connected Successfully
```

Figure 10 connected successfully of Backup Manager User Interface

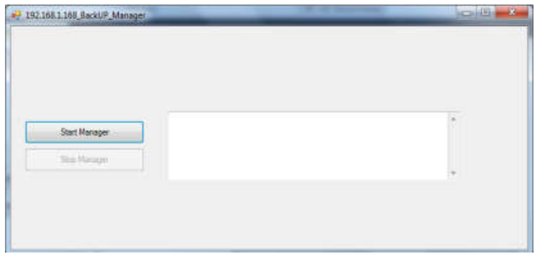
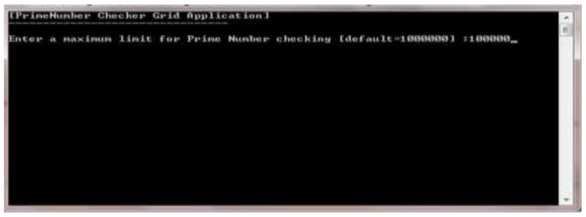


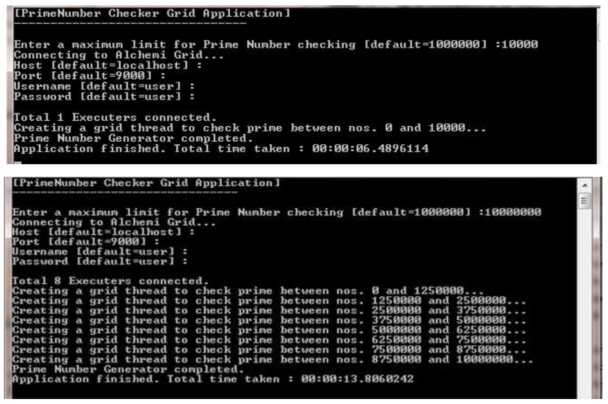
Figure 11 Backup Manager Port Scan User Interface



```
[PrimeNumber Checker Grid Application]
--------------------------------------

Enter a maximum limit for Prime Number checking [default=1000000] :100000_
```

Figure 12 Prime Number Generator Application



```
[PrimeNumber Checker Grid Application]
--------------------------------------

Enter a maximum limit for Prime Number checking [default=1000000] :10000
Connecting to Alchemi Grid...
Host [default=localhost] :
Port [default=9000] :
Username [default=user] :
Password [default=user] :

Total 1 Executers connected.
Creating a grid thread to check prime between nos. 0 and 10000...
Prime Number Generator completed.
Application finished. Total time taken : 00:00:06.4896114
```

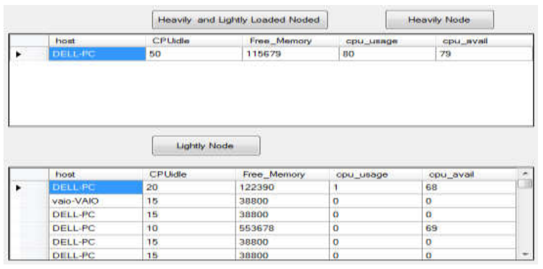
```
[PrimeNumber Checker Grid Application]
--------------------------------------

Enter a maximum limit for Prime Number checking [default=1000000] :10000000
Connecting to Alchemi Grid...
Host [default=localhost] :
Port [default=9000] :
Username [default=user] :
Password [default=user] :

Total 8 Executers connected.
Creating a grid thread to check prime between nos. 0 and 1250000...
Creating a grid thread to check prime between nos. 1250000 and 2500000...
Creating a grid thread to check prime between nos. 2500000 and 3750000...
Creating a grid thread to check prime between nos. 3750000 and 5000000...
Creating a grid thread to check prime between nos. 5000000 and 6250000...
Creating a grid thread to check prime between nos. 6250000 and 7500000...
Creating a grid thread to check prime between nos. 7500000 and 8750000...
Creating a grid thread to check prime between nos. 8750000 and 10000000...
Prime Number Generator completed.
Application finished. Total time taken : 00:00:13.8060242
```

Figure 13 Prime Number Generator Application Executions with eight and one Executors

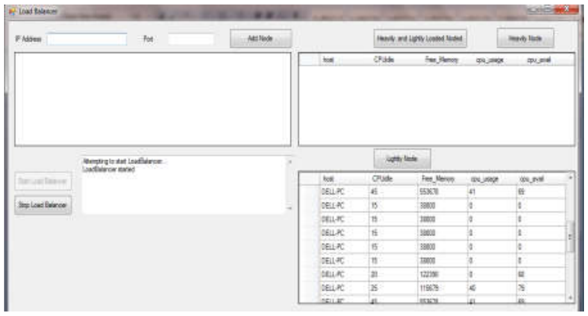
In above figures we implemented automatic backup of one computer to another computer. They executed their tasks with less time. Here Execution of Prime Number Generator Application with failure of first manager and connect with second manager. Following results are display for load balancing with fault toleranc



host	CPUidle	Free_Memory	cpu_usage	cpu_avail
DELL-PC	50	115679	80	79

host	CPUidle	Free_Memory	cpu_usage	cpu_avail
DELL-PC	20	122390	1	68
vaio-VAIO	15	38800	0	0
DELL-PC	15	38800	0	0
DELL-PC	10	553678	0	69
DELL-PC	15	38800	0	0
DELL-PC	15	38800	0	0

Figure 14 heavily loaded node and lightly loaded node

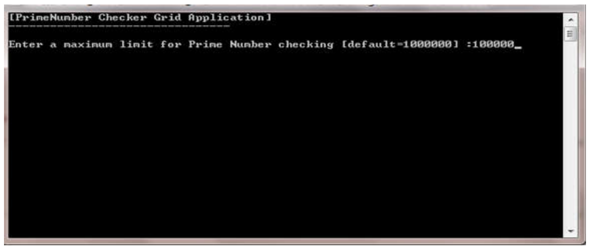


```
Attempting to start LoadBalancer...
LoadBalancer started
```

host	CPUidle	Free_Memory	cpu_usage	cpu_avail

host	CPUidle	Free_Memory	cpu_usage	cpu_avail
DELL-PC	45	553678	41	69
DELL-PC	15	38800	0	0
DELL-PC	15	38800	0	0
DELL-PC	15	38800	0	0
DELL-PC	15	38800	0	0
DELL-PC	15	38800	0	0
DELL-PC	20	122390	0	68
DELL-PC	25	115679	40	79
DELL-PC	45	553678	41	69

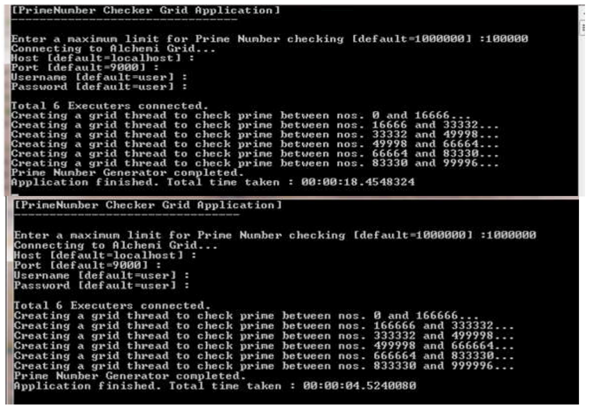
Figure 15 Transfer load from heavily loaded node to lightly loaded node



```
[PrimeNumber Checker Grid Application]
--------------------------------------

Enter a maximum limit for Prime Number checking [default=1000000] :100000_
```

Figure 16 Prime Number Generator Applications



```
[PrimeNumber Checker Grid Application]
--------------------------------------

Enter a maximum limit for Prime Number checking [default=1000000] :100000
Connecting to Alchemi Grid...
Host [default=localhost] :
Port [default=9000] :
Username [default=user] :
Password [default=user] :

Total 6 Executers connected.
Creating a grid thread to check prime between nos. 0 and 16666...
Creating a grid thread to check prime between nos. 16666 and 33332...
Creating a grid thread to check prime between nos. 33332 and 49998...
Creating a grid thread to check prime between nos. 49998 and 66664...
Creating a grid thread to check prime between nos. 66664 and 83330...
Creating a grid thread to check prime between nos. 83330 and 99996...
Prime Number Generator completed.
Application finished. Total time taken : 00:00:18.4548324
```

```
[PrimeNumber Checker Grid Application]
--------------------------------------

Enter a maximum limit for Prime Number checking [default=1000000] :1000000
Connecting to Alchemi Grid...
Host [default=localhost] :
Port [default=9000] :
Username [default=user] :
Password [default=user] :

Total 6 Executers connected.
Creating a grid thread to check prime between nos. 0 and 166666...
Creating a grid thread to check prime between nos. 166666 and 333332...
Creating a grid thread to check prime between nos. 333332 and 499998...
Creating a grid thread to check prime between nos. 499998 and 666664...
Creating a grid thread to check prime between nos. 666664 and 833330...
Creating a grid thread to check prime between nos. 833330 and 999996...
Prime Number Generator completed.
Application finished. Total time taken : 00:00:04.5240080
```

Figure 17 Prime Number Generator Application existing and proposed Executions with six Executors

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.accorfingly we adjust load and execute the application with execution time.

Comparison of existing and proposed fault tolerance and load balancing algorithm.

Input Range to find out prime number	Total No of Executer Running	Total Time to Complete execution	
		(Proposed Method)	(Existing Method)
Prime of 10000000	1	06:4896	08:0930
	2	07.0189	09.8875
	4	09.0110	11.9864
	6	11.6578	14.5436
	8	13.8060	16.8123
	10	15.8458	19.4537
	15	20.4432	26.8420

Table 1 Comparison of existing Fault tolerance and Proposed Fault tolerance

Input Range to find out prime number	Total No of Executer Running	Total Time to Complete execution	
		(Load Balancing with Fault Tolerance)	(Existing Load Balancing with Fault Tolerance)
Prime of 10000000	1	01:2896	06:1540
	2	06.0189	10.9647
	4	10.6645	15.9325
	6	04.5240	18.4548
	8	13.3265	17.9823
	10	16.5468	20.4537
	15	21.5543	27.1165

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

[1] L. Anand, D. Ghose and V. Mani, “ELISA: An Estimated Load Information Scheduling Algorithm for Distributed Computing Systems”, International Journal of Computer and Mathematics with Applications, April 1999.

[2] P. Kanungo and M. Chandwani, “A Process Migration Methodology for Distributed Computing Environment”,Indian Journal of Computing Technology, May 2006.

[3] M. Arora, S.K. Das and R. Biswas, “A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environment”, Proceedings International Conference of Parallel Processing Workshops (ICPPW "02).

[4] H. Shan, L. Oliker and R. Biswas, “Job Super Scheduler Architecture and Performance in Computational Grid Environments”, Proceedings ACM/IEEE Conference of Super Computing, Nov. 2003

[5] Kamana Sigdel, “Resource Allocation in Heterogeneous and Dynamic Networks” MS Thesis, Delft University of Technology,2005. en.wikipedia.org/wiki/Grid_computing

[6] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke , “ The physiology of the Grid” Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[7] GRMS,http://www.gridworkflow.org/snips/gridworkflow/space/GRMS

[8] Paul Townend and Jei Xu, “Fault Tolerance within a Grid Environment” Proceedings of AHM2003

[9] Anh Nguyen-Tuong, “Integrating Fault-Tolerance Techniques in Grid Applications” PhD Thesis, University of Virginia, August 2000

[10] Krishna Nadiminti, Akshay Luther, Rajkumar Buyya, “Alchemi: A .NET based Enterprise Grid System and Framework” December 2005

[11] Ferreira, L., Bieberstein, N., Berstis, V., Armstrong, J., “Introduction to Grid Computing

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

[13] Mr.Ramesh Prajapati,Dr.Samrat Khanna,” A Review Paper on Grid Computing”,IJEDR,Sep-2013

[14] Inderpreet Chopra, “Fault Tolerance in Computational Grids”, GCA'06,2006.

[15] Mr.Ramesh Prajapati,Dr.Samrat Khanna,” Grid Computing with different Fault Tolerant Mechanisms”, IJSRD,March-2014Mrs.Priya Patel,Mr.Ramesh Prajapati,Dr. Samrat Khanna,” To Improve The Performance Of Computational Grid Using Fault Tolerant And Dynamic Load Balancing Algorithm For Grid Environment”, IJTRE ,May-2016

[16] E. Beguelin, Seligman and P. Stephan, “Application Level Fault Tolerance in Heterogeneous Networks of Workstations”, Journal of Parallel and Distributed Computing on Workstation Clusters andNetworked-based Computing, Vol. 43(2), 1997, pp. 147–155.