TO IMPROVE THE PERFORMANCE OF COMPUTATIONAL GRID USING FAULT TOLERANT AND DYNAMIC LOAD BALANCING ALGORITHM FOR GRID ENVIRONMENT

Priya Patel[1], Mr.Ramesh Prajapati[2], Dr. Samrat Khanna[3]
[1]Department of Computer Engineering, SCET, Saij, Kalol, Gujarat, India
[2]Rai University, Center for Research & Development Saroda, Dholka, India
[3]Dept. of Information Technology, Istar, Sardar patel centre for science & Tech., V.Vnagar, India

ABSTRACTL: Grid computing provides service of sharing data storage capacity and computer power over the Internet. Resource management scenarios often include resource discovery, resource monitoring, resource inventories, resource provisioning, fault isolation, variety of autonomic capabilities and service level management activities. Out of these scenarios, fault tolerance and Load Balancing are main research areas. In this paper First issue our main focus is on the development of fault tolerance system for computational grids and applies the Load Balancing. For this we had setup a computational grid based on the Alchemi middleware. In case of failure of the central manager, backup manager will take its control and avoids the grid to fail. This makes load balancing and fault tolerance more important in case of computing grid. This paper second issue introduces an algorithm which balances the load among the resources and also increases the reliability of the grid environment. The main goal of load balancing is to provide a distributed, low cost, scheme that balances the load across all the processors.

Keywords: Grid computing, Grid Architecture, Load balancing, Fault Tolerance.

I. INTRODUCTION

Grid is a system that coordinates resources that are not subject to centralized control using standard, open, general-purpose interfaces and protocols to deliver non-trivial qualities of service. A grid is a type of parallel and distributed system that enables the allocation, selection and aggregation of resources distributed across multiple administrative domains based on their (resources) availability, capacity, performance, cost and quality of service requirements. A grid is a collection of machine sometimes referred to as nodes, resources, donors, members, clients, hosts, engines and many other such terms [2]. Grid computing is enabled by relatively high-performance computers, robust computer networks, grid management software, and the divisibility of difficult scientific problems. In Grid computing, individual users can access computers and data, transparently, without having to consider location, operating system, account administration, and other details. Grids tend to be more loosely coupled, heterogeneous, and geographically distributed. In Grid computing details are abstracted, and the resources are virtualized [1].

II. LITERATURE SURVEY

FAULT TOLERANCE:

As a result of the complex nature of heterogeneous networks, fault tolerance [5] is a major concern for the network administrators, and there are various ways that detection of such occurrences can be accomplished. When a fault occurs, it is important to: rapidly determine exactly where the fault is, Isolate the rest of the network from the failure so that it can continue to function without interference, Reconfigure or modify the network in such a way as to minimize the impact of operation without the failed component or components, and repair or replace the failed components to restore the network to its initial state.

A. Function of Fault Tolerance

The fault tolerance is "to preserve the delivery of expected services despite the presence of fault-caused errors[6] 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 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. For making fault tolerance system a success, we must consider:

- How new nodes join the system,
- How computing resources are shared,
- How the resources are managed and distributed

B. Fault Tree Analysis

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 users. The fault tree analysis[10] classifies faults that may take place in Grid Computing. In the figure various kinds of faults that can occur have been shown. There are mainly six classes of faults as discussed below:

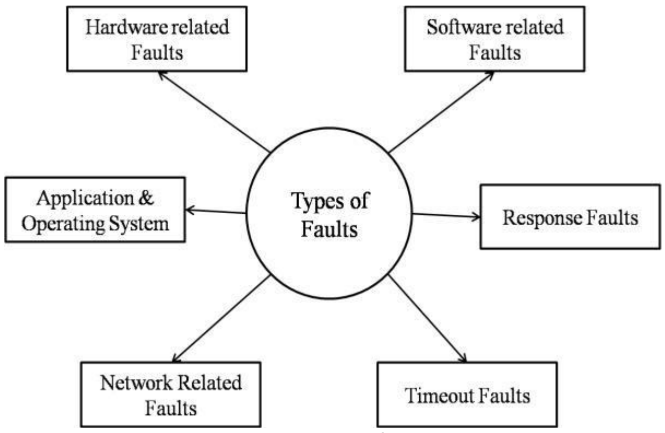


Figure 1 Type of Faults

LOAD BALANCING:

Load balancing is a technique to enhance resources, utilizing parallelism, exploiting throughput improvisation, and to cut response time through an appropriate distribution of the applications [9]. To minimize the decision time is one of the objectives for load balancing which has yet not been achieved. Job migration is the only efficient way to guarantee that submitted jobs are completed reliably and efficiently in case of process failure, processor failure, node crash, network failure, system performance degradation, communication delay; addition of new machines dynamically even though a resource failure occurs which changes the distributed environment [11].

As following Figure 2 load balancing feature can prove invaluable for handling occasional peak loads of activity in parts of a larger organization. These are important issues in Load Balancing:

An unexpected peak can be routed to relatively idle machines in the Grid.

If the Grid is already fully utilized, the lowest priority work being performed on the Grid can be temporarily suspended or even cancelled and performed again later to make room for the higher priority work.

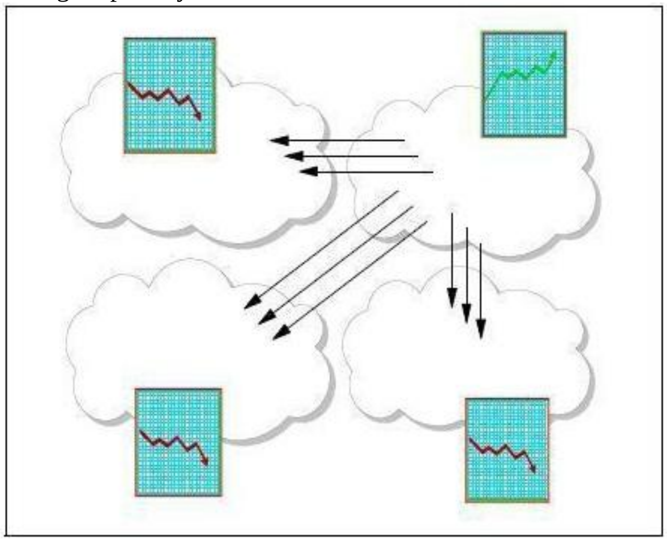


Figure 2 Job Migration [5]

III. PROBLEM STATEMENT

In this paper First issue our main focus is on the development of fault tolerance system for computational grids. For this we had setup a computational grid based on the Alchemi middleware. Alchemi is a .NET-based grid computing framework that provides the Runtime machinery and programming environment required to construct computational grid. In case of failure of the central manager, backup manager will take its control and avoids the grid to fail. In grid computing the resources are shared and dynamic in nature, which affects application performance. To improve the global throughput of these environments, effective and efficient load balancing algorithms are fundamentally important. Load Balancing is one of the most important factors which can affect the performance of the grid application. Main purpose of load balancing is to analyze problems due to which load balancing is required in grid computing and to transfer heavy loaded nodes to lightly loaded nodes based on job migration policy.

IV. PROPOSED WORK

PROPOSING THE BACKUP MANAGER

While working with the setup grid, the failure of the centralized manager causes the whole grid to hangs down. Till the manager doesn’t restart, the grid remains inaccessible. So to avoid such kind of grid failure, we have introduced the concept of the backup manager. The backup manager also uses the heart beat phenomenon to query the status of manager. The steps for implementing the backup manager are:

After every heartbeat interval, the leader node sends a packet to the backup manager. Figure 3 shows the Backup manager concept. Packet received by the backup manager contains information about each of the nodes in the group and its arrival indicates that the leader is up and running. The backup leader updates its database using the data obtained from the received packet. In case of absence of packet for certain predefined time interval lets the backup manager to consider the master has failed, and it itself takes the control as new master. This change is multicast to the executors and they update their connections database in order to accept Backup Manager as their new master. Figure 3 shows the new connection established between the executors and the backup manager. We try to use these steps for providing the support for backup manager for Alchemi based computational grids. From there the backup manager accesses the database to control the grid.

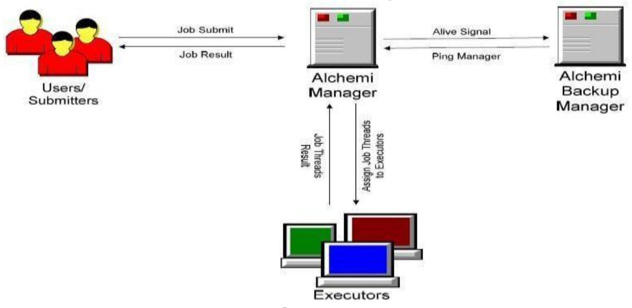


Figure 3: Backup Manager Concept

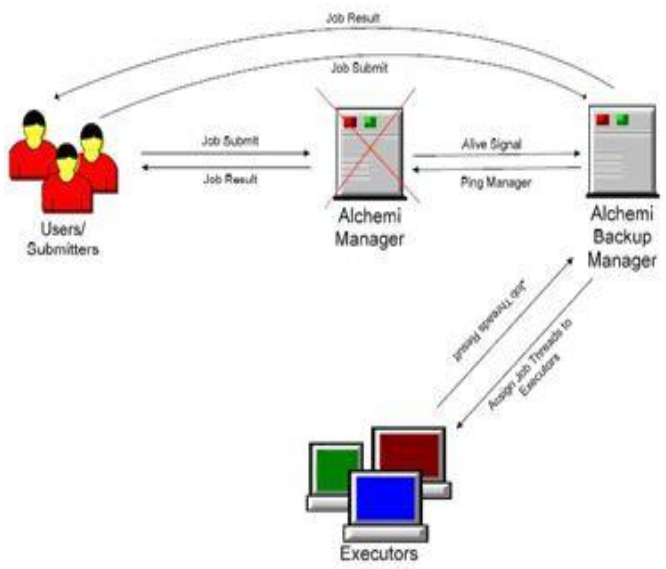


Figure 4 Failure of Manager Leads Backup Manager to take the control

LOAD BALANCING

In this paper we propose a dynamic load balancing algorithm for improving performance of grid computing. Here we mention are four steps: load monitoring, synchronization, rebalancing criteria job migration. In proposed load balancing algorithm the activities can be categorized as following:

Arrival of any new job and queuing of that job to any particular node, Completion of execution of any job, Arrival of any new resource, Withdrawal of any existing resource.

Code for to algorithm: Function: LoadBalancing_start Return Type:

Boolean Start:

If (CPU Idle and Free Memory of Node is Min and Queue Length is Max)

HeavilyLoaded_Node

HeavilyLoaded_list= HeavilyLoaded_list + 1 (new selected node);

End if

If (CPU Idle and Free Memory of Node is Max and Queue Length is Min)

Lightlyloded_Node

End if

Migrate Heavy Loded_Node_Job to Lightly Loded_Node

End

LoadBalancing_start (): this function also return Boolean value. If on the basis of given parameters

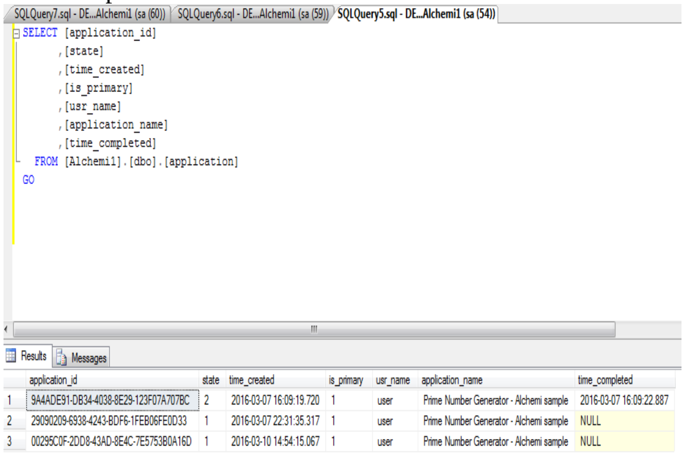
(CPU utilization and queue length) load balancing will be required it will return true else it will return false. This function also updates two lists: HeavilyLoaded_list and LightlyLoaded_list.

	Information Policy	Firing Triggering	Hitting Selection
Existing Load Balancing	information is collected using periodic approach	Triggered based on Queue Length	Task is selected for Migration using Job Length as criteria.
Proposed Load Balancing	information is collected using Activity based Approach	Triggered based on Queue Length and current CPU Load	Task is selected for Migration based upon CPU consumption of tasks

Main difference between existing Load Balancing algorithm and proposed Load Balancing is in implementation of three policies: Information Policy, Triggering Policy and Selection Policy. For implementation of Information Policy all existing Load Balancing algorithm use periodic approach, which is time consuming. The proposed approach uses activity based approach for implementing Information policy. For Triggering Load Balancing proposed algorithm uses two parameters which decide Load Index. On the basis of Load Index Load Balancer decide to activate Load Balancing process. For implementation of Selection Policy Proposed algorithm uses Job length as a parameter, which can be used more reliably to make decision about selection of job for migration from heavily loaded node to lightly loaded node.

V. IMPLEMENTATION STRATEGY AND RESULTS

Create Replication Database

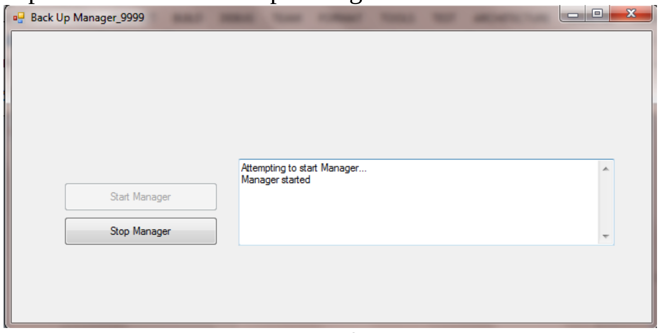

SQLQuery7.sql - DE...Alchemi1 (sa (60))
SQLQuery6.sql - DE...Alchemi1 (sa (59))
SQLQuery5.sql - DE...Alchemi1 (sa (54))
Results
Messages

```
SELECT [application_id]
      ,[state]
      ,[time_created]
      ,[is_primary]
      ,[usr_name]
      ,[application_name]
      ,[time_completed]
  FROM [Alchemi1].[dbo].[application]
GO
```

```
   application_id                        state  time_created             is_primary  usr_name  application_name                        time_completed
1  9A4ADE91-DB34-4038-8E29-123F07A707BC  2      2016-03-07 16:09:19.720  1           user      Prime Number Generator - Alchemi sample  2016-03-07 16:09:22.887
2  29090209-6938-4243-BDF6-1FEB06FE0D33  1      2016-03-07 22:31:35.317  1           user      Prime Number Generator - Alchemi sample  NULL
3  00295C0F-2DD8-43AD-8E4C-7E5753B0A16D  1      2016-03-10 14:54:15.067  1           user      Prime Number Generator - Alchemi sample  NULL
```

Figure 5 Screenshot of Running Application

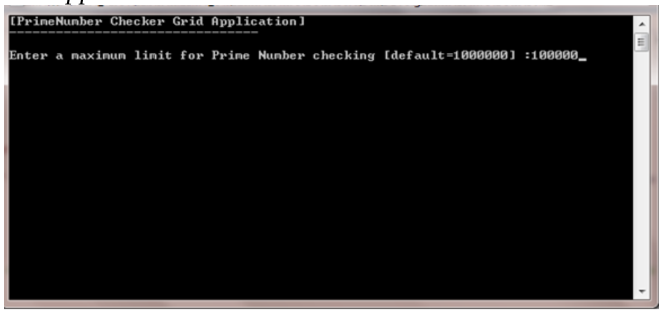
Implementation of Backup manager



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

Figure 6 Screenshot of Backup Manager

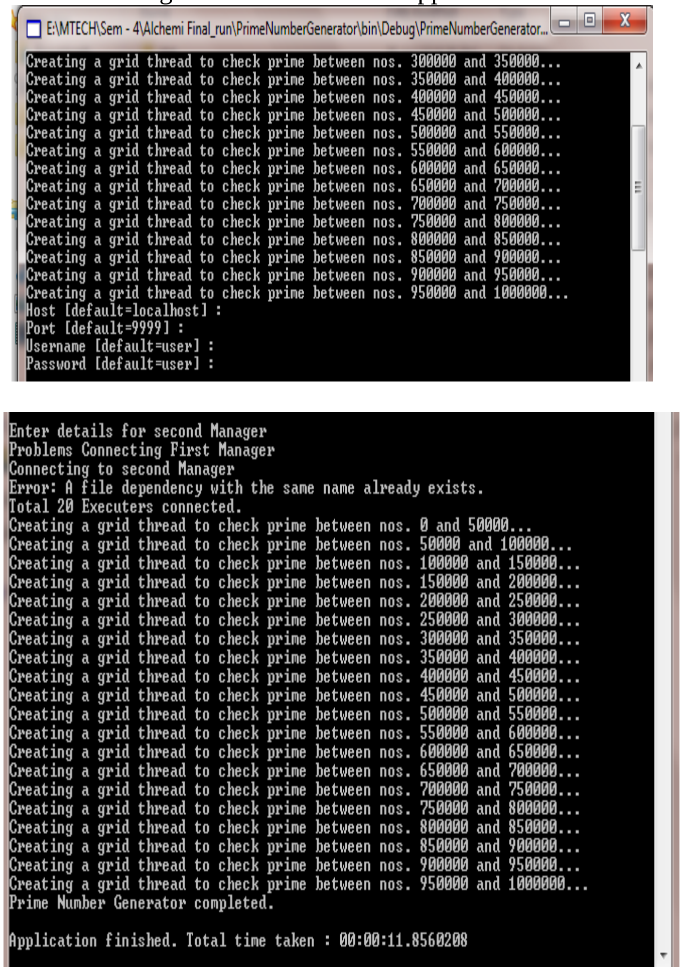
Run Application



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

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

Figure 7 Prime Number Application


E:\MTECH\Sem - 4\Alchemi Final_run\PrimeNumberGenerator\bin\Debug\PrimeNumberGenerator...

```
Creating a grid thread to check prime between nos. 300000 and 350000...
Creating a grid thread to check prime between nos. 350000 and 400000...
Creating a grid thread to check prime between nos. 400000 and 450000...
Creating a grid thread to check prime between nos. 450000 and 500000...
Creating a grid thread to check prime between nos. 500000 and 550000...
Creating a grid thread to check prime between nos. 550000 and 600000...
Creating a grid thread to check prime between nos. 600000 and 650000...
Creating a grid thread to check prime between nos. 650000 and 700000...
Creating a grid thread to check prime between nos. 700000 and 750000...
Creating a grid thread to check prime between nos. 750000 and 800000...
Creating a grid thread to check prime between nos. 800000 and 850000...
Creating a grid thread to check prime between nos. 850000 and 900000...
Creating a grid thread to check prime between nos. 900000 and 950000...
Creating a grid thread to check prime between nos. 950000 and 1000000...
Host [default=localhost] :
Port [default=9999] :
Username [default=user] :
Password [default=user] :
```

```
Enter details for second Manager
Problems Connecting First Manager
Connecting to second Manager
Error: A file dependency with the same name already exists.
Total 20 Executers connected.
Creating a grid thread to check prime between nos. 0 and 50000...
Creating a grid thread to check prime between nos. 50000 and 100000...
Creating a grid thread to check prime between nos. 100000 and 150000...
Creating a grid thread to check prime between nos. 150000 and 200000...
Creating a grid thread to check prime between nos. 200000 and 250000...
Creating a grid thread to check prime between nos. 250000 and 300000...
Creating a grid thread to check prime between nos. 300000 and 350000...
Creating a grid thread to check prime between nos. 350000 and 400000...
Creating a grid thread to check prime between nos. 400000 and 450000...
Creating a grid thread to check prime between nos. 450000 and 500000...
Creating a grid thread to check prime between nos. 500000 and 550000...
Creating a grid thread to check prime between nos. 550000 and 600000...
Creating a grid thread to check prime between nos. 600000 and 650000...
Creating a grid thread to check prime between nos. 650000 and 700000...
Creating a grid thread to check prime between nos. 700000 and 750000...
Creating a grid thread to check prime between nos. 750000 and 800000...
Creating a grid thread to check prime between nos. 800000 and 850000...
Creating a grid thread to check prime between nos. 850000 and 900000...
Creating a grid thread to check prime between nos. 900000 and 950000...
Creating a grid thread to check prime between nos. 950000 and 1000000...
Prime Number Generator completed.

Application finished. Total time taken : 00:00:11.8560208
```

Figure 8 manager fail and connect to backup manager

Experiment results

Input Range to find out prime number	Total No of Executer Running	Total Time to Complete execution	
		(In Second)(BM Application)	(In Minute)(Existing BM Application)
Prime of 1000000	2	00:04:0872	01:04:0944
	3	00:04:0404	01:03:0637
	4	00:04:4460	01:01:0265
	6	00:04:5240	01:00:0101
	8	00:09:6096	01:06:0859
Prime of 10000000	2	00:04:0900	01:05:0945
	4	00:04:0600	01:04:0400
	6	00:04:6240	01:00:0010
	8	00:13:8060	01:09:7682

Table 1 comparison of Different Number of Executer with Different input range

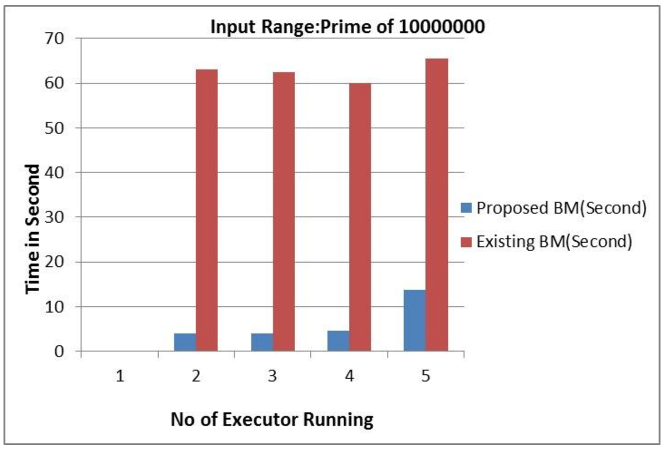


Figure 9 Comparison of no of executor running with respect to time

VI. CONCLUSIONS AND FUTURE ENHANCEMENTS

In this research I analyzed existing Load Balancing algorithm and proposed an enhanced algorithm which more efficiently implements three out of five policies implemented in existing Load Balancing algorithm.I studied the problems and challenges included with faults in computational grid and proposed algorithm use for finding the various kind of faults based on Alchemi. It also calculated the efficiency of our proposed system under various situations. Future work includes, find optimal approach for better performance of applications running in grid. And also we will try to achieve more efficiency with new police for load balancing. Furthermore, the proposed algorithm can also be replaced with the existing work to improve the performance of grid.

RERERENCES

[1] AmitKapoor,Meenaks Gupta, Vasudha Rani "The Impact of Load Balacing on Grid Computing Performance-review"May 2011.
[2] Mr.Gaurav Sharma, JagjitKaur Bhatia "A Review on Different Approaches for Load Balancing in Computational Grid" by JGRCS April 2013.
[3] Paritosh Kumar, Dr. InderveerChana, "Load Balancing and Job Migration in Grid Environment" Thesis,In computer science and engineering department, thapar university, July 2009.
[4] ManikMujumdar, MeenakshiBheevgadev, Latesh Malik, Dr.RajendraPatrikar, "High Performance Computational Grid –Fault Tolerance at System Level" 2008 IEEE.
[5] AbhayTripathi, "A Novel Load Balancing Algorithm in Grid Computing", JAN-FAB 2014.
[6] Ratnesh Kumar Nath, Ms. InderveerChana, Dr. (Mrs.) SeemaBawa, "Efficient Load Balancing Algorithm in Grid Environment", ThaparUniversity,May 2007.
[7] K JairamNaik,Dr A Jagan, "A novel algorithm for fault tolerant job Scheduling and load balancing in Grid Computing environment",IEEE 2015.
[8] S.K.Karthikumar, M.UdhayaPreethi, "Fair Scheduling Approch For Load Balancing and Fault Tolerant in Grid Environment", 2013 IEEE.
[9] http://www.gridcomputing.com.
[10] Sumant Jain, jyotiChaudhary, "New Fault Tolerant Scheduling Algorithm Implemented using Check Pointing in Grid Computing Envionment",2014 IEEE.
[11] InderpreetChopra,"Fault Tolerance in Computational Grids", GCA'06, 2006.
[12] Mr.RameshPrajapati "Fault Tolerance Mechanism for Computational Grid Using Checkpoint Algorithm", IJEDR, 2013.
[13] MangeshBalpande,UrmilaShrawankar,"Robust fault Tolerant job scheduling Approach in Grid Environment",IEEE 2014
[14] SumantJain,JyotiChaudhary,"New Fault Tolerant scheduling Algorithm implemented using check Pointing n Grid Computing Environment",IEEE 2014
[15] K.Nirmaladevi ,A.Tamilarasi, "Dynamic Scheduling in Grid environment with the improvement of Fault tolerant level ",IEEE ,April 2015
[16] S.Gokuldev, ShahnaMoideen, Associate Professor in Computer of science and Engineering, "Global Load Balancing and Fault Tolerant Scheduling in Computational Grid", Published in IJEIT, May 2013.
[17] PrakashKumar,PradeepKumar,Vikas Kumar, "An Effective Dynamic Load Balacing Algorithm for Grid System" IJETT, August 2013.
[18] R.Manimala,P.Suresh, "Load balancing job Scheduling Approach for Grid Environment", IEEE 2015.
[19] Ankita Jindal RK Bansal,SavinaBansal "Efficint Load Balancing Scheduling for Deadline Constrained Tasks on Grid Computing" IEEE.
[20] AnkitPunia, Ms.Pooja Mittal, "A Review: Grid Computing", IJCSMC, April 2014.
[21] Prakash Kumar, Pradeep Kumar, Vikas Kumar, "Computational Grid System Load Balancing Using an Efficient Scheduling Technique", IJCTT, August 2013.
[22] Jagdishchandrapatni, Dr. M. S. Aswal "Load balancing Strategies for Grid Computing" IEEE, 2011.
[23] Dr.D.V.SubbaRao, "Automatic checkpointing based on Fault Tolerance in Computational Grid", IEEE 2014.
[24] P. Latchoumy1 and P. Sheik Abdul Khader "Survey on Fault Tolerance in Grid Computing", Ijcses 2011