IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

Evaluation of Fault Tolerant & Min-Max Load Balancing Algorithm in Grid Computing

Mahesh Reddy G[1*], Lakshminarayana G[2], Srinuvasa Reddy K[3]

[1*, 2,3]Department of CSE, Kallam Haranadhareddy Institute of Technology, JNTUK, Guntur, A.P, India

[*]*Corresponding Author: mahesh.gogula@gmail.com Mobile: +91- 9505276512*

Available online at: www.ijcseonline.org

Accepted: 19/Jul/2018, Published: 31/Jul/2018

Abstract- Grid computing is a growing technology, which have hundreds or thousands of computational nodes to execute large applications. The main issues consider in grid computing is Load balancing, Fault tolerant and Fault recovery. In order to utilize the resource efficiently and to satisfy all the requirements of the user, there is need for effective scheduling algorithm. Scheduler schedule job to available resources in the sites and result submit to user. Otherwise if any faults detects by fault detector, In the event of failures, how to execute the job from the processor failure. Since grid is more difficult, complex to implement and manage, so there could be the differences in their performance under diverse experimental conditions. We should modify the algorithm. To achieve high throughput and proper resource utilization we purpose Min-Max fair scheduling algorithm for load balancing. This paper focus on how to increase performance of Grid Environment. The proposed system builds on using Grid Simulation Toolkit (Gridsim).

Keywords: Load Balancing, Grid Computing, Fault Tolerant & Fault Recovery, Min-Max Fair Scheduling algorithm

I. Introduction

Grid computing is basically taking number of inexpensive personal computers and connecting them via network to build a super computer which can utilize the idle processing time. The notion is that computational grids will offers users the ability to connect to network and make use of computing resources such as mass storage and large processing capabilities.

In grid, the scheduling process is done by the grid scheduler. Grid scheduler mainly deals with how efficiently and appropriately to assign resources to jobs. While choosing resources for jobs, scheduler should consider various characteristics of job such as length of job, user deadline and resource characteristics such as capability, communication time and cost. The scheduler will monitor all the resources by a resource monitoring process before dispatching the job the scheduler. The grid schedulers receive the job from the client and assign them to the particular resource. If any job failed due to some in the sites it should be tolerated. So we can purpose Fault tolerance. At the primary site, a backup is takes from each job. If any primary failed, the backup always succeeds.

Two types of load balancing policy in grid environment are static and dynamic load balancing. Static is not suitable to grid computing (change the load with respect time). Based on the load at the time it allocates the job to resource and also work stations are not constantly monitor. In dynamic load balancing, work stations are constantly monitored and selection of selection of Workload done at run time and also uses the current load information for decision making. Dynamic load balancing algorithm will give better performance then static load balancing. Each site in the grid environment will provide its hardware information; time and how many resources are available in each site in the grid environment. That information is recorded for decision making purpose. The grid scheduler is a manager it will manage the state of sites to execute the jobs. As arriving jobs are placed in job queue, the load of the system is increased with increase the length of job queue. While load in system is balanced, the grid scheduler will receive the job from the job queue and perform the scheduling process.

The remainder of this paper is organized as follows:

Section 1. Introduction of Grid Computing
Section 2. Literature Survey
Section 3. Implementation
Section 4. Min-Max Fair Scheduling Algorithm
Section 5. Results and Analysis
Section 6. Conclusion.

II. Previous Work

In the grid the load balancing algorithm will follow three polices there are information policy, transfer policy, location policy and selection policy. The information policy specifies what workload information to be collected, when it to be collected and from where it to

be collected. The resource monitoring system will monitor the status of the resource based on the resource load information the transfer policy will decide whether the resource have eligibility to act as s sender or receiver. Sender means it will transfer the job to the resource. Resource means it will receive the job from another resource. Based on the transfer policy the location policy will select a suitable partner for sender or receiver. If the resource is an eligible sender the location policy seeks out an eligible receiver to transfer the job. Suppose the resource is an eligible receiver location policy seeks out an eligible sender. Once we decide the resource as an eligible sender or receiver the selection policy. The selection policy defines which job should be migrated from heavily loaded to lightly loaded node. [16]

An extended version of the mutual information feedback policy has been proposed to achieve average response times of jobs. [3] Jobs will fail when the site where they are located fails due to hardware faults. The faults can be transient or permanent and are assumed to be independent. For each job, the backup is scheduled after its primary. There exists a fault detection mechanism such as fail-signal and acceptance test to detect processor and job failure. [13]With respect to fault tolerance, primary backup approach proposed. Where one server is selected as the primary and all the other are backups. If the primary fails one of the backups takes over. In this paper the backup is scheduled for each primary. [14]

In this article, an approach for load balancing and scheduling is proposed. The throughput and performance of the grid environment will greatly improve by fair scheduling approach with equal opportunity to all jobs is designed. The fair scheduling approach follows hybrid scheduling by calculating residue value for each job for number of iterations until the residue gets to zero. This approach is linear and iterative in nature which eliminates the fluctuations in the response time. [10]

III. Proposed Work

1. Globus Toolkit (Grid Simulation Toolkit):

The Globus Toolkit is a set of services and software libraries that support Grids and Grid applications. The Toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection and portability. It is packaged as a set of components that can be used either independently or together to develop useful Grid applications and programming tools. Globus Toolkit components include Grid Security Infrastructure (GSI), which provides a single-sign-on, run-anywhere authentication service, with support for delegation of credentials to sub components, local control over authentication service and mapping from global to local use identifies; the Grid Resource Access and Management (GRAM) protocol and service, which provides remote resource allocation and process creation, monitoring and management services; the Meta computing Directory Service (MDS), an extensible Grid Information service that provides a uniform framework for discovering and accessing system configuration and status information such as computer server configuration, network status or the locations of replicated datasets and GridFTP, a high-speed data movement protocol. The proposed system builds on using Grid Simulation Toolkit Version 5.

2. System Model

In our simulation model, each site consists of one machine and each machine consists of one or more processors. As shown in Fig. 1, the environment has a global and local grid scheduler which is a software component that resides within each site. The grid scheduler will manage the sites, provide the resources, receive the job from the client and assign them to the processor in the grid environment. If the local/remote site wants to join in the grid environment, then resource information like computing speed, storage, communication bandwidth etc., should be informed to the GIS.

System must satisfy following conditions:

- The grid scheduler administers the state of sites that execute the job request.
- Future decision making should be made based on the contribution of the resource to the previous request with time stamp.
- The user job request must contain the information about the minimum required resource units.

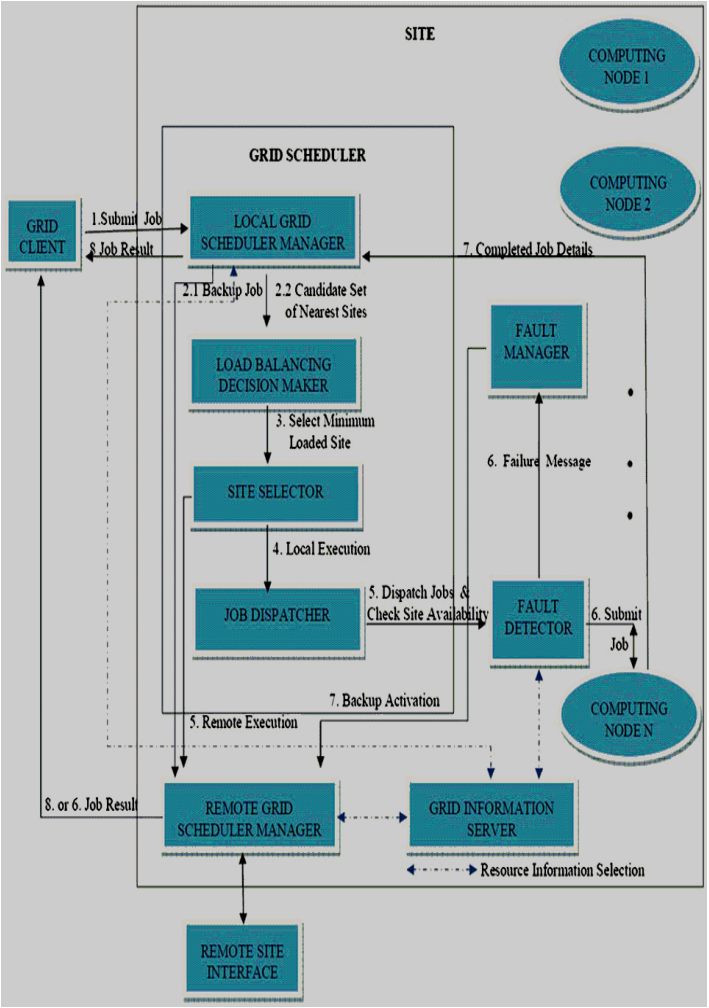


Fig. 1. Process flow of Grid Scheduler

3. Grid Model:

Grid environment consists of n sites connected to simple network, labeled as s_1, s_2, s_3... s_n. We also assume that there exists a well known set of brokers (Scheduler), denoted as b_1... b_h. Each broker is located in certain grid site. We use home site (b_i) to denote the site that the broker b_i resides currently. Each broker is responsible for exchanging information with other brokers and forwarding the lightest loaded site from remote region to the heaviest loaded site in the current site.

4. Proposed Approach Framework:

Initially the grid client will submit the job to the grid scheduler. Grid scheduler works in three phases: Resource Discovery, Resource Allocation and Job Execution. Resource Discovery phase involves identifying the available resources from resource pool. Whereas resource allocation phase involves selection of suitable resource and allocating the selected resource to the jobs. Third phase is executing the jobs at resource locations. The grid scheduler will receive job from the
user and place the job in queue. The scheduler will assign the jobs to appropriate sites. The grid scheduler backup the received job and discover the candidate set of nearest sites and give the job to the load balancing decision maker. The load balancing decision maker will select the minimum loaded site and give the job to the site selector.

The site selector will decide whether the jobs are to be executed in the local or remote area. The job dispatcher will dispatch the jobs based on the availability of the sites. The fault detector will monitor the computing nodes. If the computing nodes get failed then fault detector will send the failure message to the fault manager.

5. Application Design:

This approach follows decentralized strategy which implies that any task in grid site may utilize any available resource. In dynamic decentralized strategy, decisions are made at any time. The grid scheduler performs state collection, decision making and dispatch o task to the available site. The fault detector will monitor computing nodes. If the computing node completes the job successfully then grid scheduler will send the completed job details to the grid user. Another wise failure job details send to the grid user.

6. Determining the List of Candidate Keys:

Each site in the grid environment has more chances to enter in to the busy state because of grid environment is a dynamic. So selecting site is more important.

An array V_i is proposed to evaluate the

$$Vj = \sum_{i}^{n} Wi * f(Xij), \mathbf{1 \le j \le N}$$

Where

i-which site in the Grid Environment

j- Computing Node

$\mathbf{W_i}$- weight value for each site in Grid computing so we are taking it as MIPS Rating of each site in Grid Environment.

$\mathbf{F(X_{ij})}$- Free Resources in each site

$\mathbf{V_j}$- Speed of execution rate for each site

In Fig 2, based up on the value of W_i, job is allocated to resource. Grid environment contain sites, each site has more number of resource. MIPS value changed to each site so If a job is allocate to resource based up on the value of MIPS choose the
minimum loaded site i.e minimum MIPS value of site and allocate the job to that site first, so increase the execution speed And proper resource utilization is done. In this we compare the FCFS (FIRST COME FIRST SERVE) & FAIR SCHEDULING based up on the MIPS values so we can get better results with Min-Max fair scheduling algorithm.

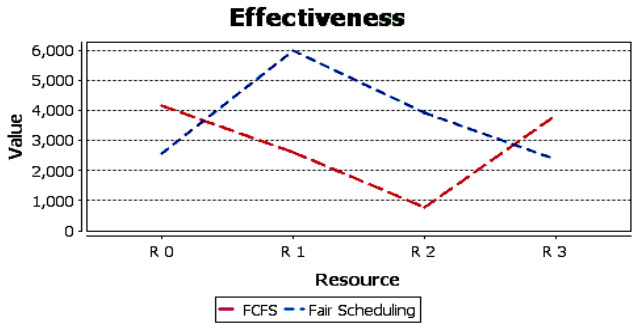


Fig. 2. Efficiency of Resources.

In Fig. 2, R0, R1, R2 and R3 are resources and their MIPS values (Speed of the execution) taking randomly. R3 has Minimum MIPS value compare to another resources and R0 has Minimum MIPS value compare to remain resources and R2 has minimum MIPS value compare to R1 this process done up to last resource i.e. R3<R0<R2<R1 so we first allocate the job to R3 and next R0 similarly up to R1. So we get the proper resource utilization and then performance of the grid also increase.

7. Job Allocation:

Suppose Grid environment have 4 Gridlet sites i.e. (Gridlet0, Gridlet1, and Gridlet2 & Gridlet3) and resources are R0, R1, R2 and R3 and jobs are job0, Job1, Job2 and Job3 respectively. From Fig. 3, Jobs are allocated in this manner.

Gridlet1 (Job1) is submitted to site Res3 and ID: 17
Gridlet3 (Job3) is submitted to site Res0 and ID: 5
Gridlet0 (Job0) is submitted to site Res2 and ID: 13
Gridlet2 (Job2) is submitted to site Res1 and ID: 9

Here job1 submitted to R3 because job 1 requires more demand rates compare to another jobs. Based up on the demand rates of jobs, jobs are allocated to minimum loaded site. Demand rates of jobs (Job1>Job3>Job0>Job2)

8. Site selection

We have to calculate the list of candidate keys for each site in grid environment. Those values are stored in V_j. Values are compared to each other if the value is highest than others so we can select the highest value corresponding site as site selection. So job first allocated to this site first. Here V_j is called effectiveness of site (OR) threshold value. The sites which are within the threshold value are considering as effective sites and stored in ascending order.

9. Fault Detection & Fault tolerant:

At first jobs are waited in the queue and then according to fair rates then jobs are assigned to the resources. The resource gets starts to execute the jobs. Most of the jobs are completed at correct time. If any failure occurred i.e. resource does not execute the job in it. The fault detector fined it and send message to the fault manager. The fault manager will execute the secondary of the job in another resource. Now this resource will complete the job successfully.

10. Min-Max Fair Scheduling Algorithm:

We propose new Algorithm for distribution of the load in Grid Environment. Here inputs and outputs to Grid Environment are mentioned in below algorithm. Based upon the algorithm, Processor Capacity is allocated to the jobs.

Algorithm

Input-No. Of Grid Environments, Users and Jobs
Output-All Jobs are successfully executed
Begin
Calculate CPU Capacity(C) and assign Demand Rates (D) and weights to jobs (W)
Calculate Residue= Divide Processor Capacity by total Weights of jobs
Start First Iteration
If (W==1)
Allocate Residue to jobs
Else If (W==2)
Allocate Double Residue to Jobs
End

$$Calculate\ Residue = C - \sum_{k=0}^{n} FirstIteration(k)$$

Residue>0
End
Start Second Iteration
Begin
Calculate residue=Divide Remain Processor Capacity by Total Remain Weights of jobs
If (W==1)
Allocate Residue to Jobs
Else If (W==2)
Allocate Double Residue to Jobs
Calculate Residue
If (Residue==0)
End

Each task has different demand rates and weights. According to the weight the tasks are scheduled in the grid environment. In grid environment, processor having the capacity limits. The total weight of the tasks are calculates. And then divide the processor capacity by the total weight of the task. The calculated units are assigned to the tasks having normal priority. The residue of the first iteration is calculated. The residue is divided by the total weight of remaining tasks. The calculated units are assigned to the task having normal priority and then twice the calculated units are assigned to the task having high priority. Likewise the capacity of the processor is shared or scheduled to the tasks.

In Table 1, we take 6 jobs with id 0,1, 2, 3, 4, and 5. Assign demand rates 5, 84, 3, 40, 5 and 20 and weights 1, 1, 1, 2, 1, 2 respectively. Total capacity=60. Divide capacity 60 by total weights (60/8=7.5) (i.e. we can divide 60 into 8 parts).In First Iteration, Job 0, 1, 2 and 4 of weights are equal to 1 then residue is assign to job 0, 1, 2 and 4. In allocated resources, job 0 demand rates are 5 allocated & residue value is also allocated to job 1, job 2 & job 4. Job 1 has demand rate 84, in that 7.5 allocated remain rates are allocated in next iteration. Job 2 has 3 and assign resources are 7.5 in that 3 resources are allocated to job 2. Job 4 has demand rates 5; allocated resources are 7.5 in that 5 resources are allocated. Job 3 and job 5 assign weights 2 and demand rates 40 and 20 respectively. Double the residue value is allocated to job 3 and job 5. I.e. 2*7.5=15; 15 is allocated to job 3 and job 4. In first Iteration, job 0, 2 and 4 are completed. Calculate the residue value= (Total capacity-allocated capacity) i.e. residue value=60-50.5=9.5 In Second Iteration, residue value is allocated remain jobs. Remain total jobs of weights 5(1+2+2); 9.5 divided with 5 and gets 1.9. This value is allocated to job 1, job 2 and job 5. Job 1 has weight 1 so 1.9 value is allocated. So job 1 gets 9.4 resources in second iteration. Job 3 and job 5 has weights 2 than double the residue value and allocated to Job 3 and Job 5 i.e. 2 * 1.9 =3.8. Residue 3.8 value is allocated to Job 3 and Job 5 then Job 3 got 18.8 and Job 5 got 18.8 and residues value gets to zero. So this way shared processor capacity is fully utilized.

Table 1. Min-Max Fair Scheduling

Job id	Demand rates	Weights	First iteration	Second iteration
0	5	1	5	5
1	84	1	7.5	9.4
2	3	1	3	3
3	40	2	15	18.8
4	5	1	5	5
5	20	2	15	18.8

IV. Results and Analysis

In this section, we show the results of my algorithm. In this results, how to create grid Environment and sites and resources and how to assign jobs to resources and how to schedule the resources to execute the jobs. In the results, Girdlet0, Gridlet1, Gridlet3 are successfully executed but Gridlet2 is failed so we can assign failed jobs to another Gridlet4. Job 2 is submitted to Res 4 and successfully executes jobs and display recovery done to user.

RESULTS:

---Enter Number of sites of grid Environment: 4

Creating a gridlet with name =gridlet_0 and ID =0
Creating a gridlet with name =gridlet_1 and ID =1
Creating a gridlet with name =gridlet_2 and ID =2
Creating a gridlet with name =gridlet_3 and ID =3

---Enter No of user's for Grid Environment: 4

Creating a griduser with name =user_0
Creating a griduser with name =user_1
Creating a griduser with name =user_2
Creating a griduser with name =user_3

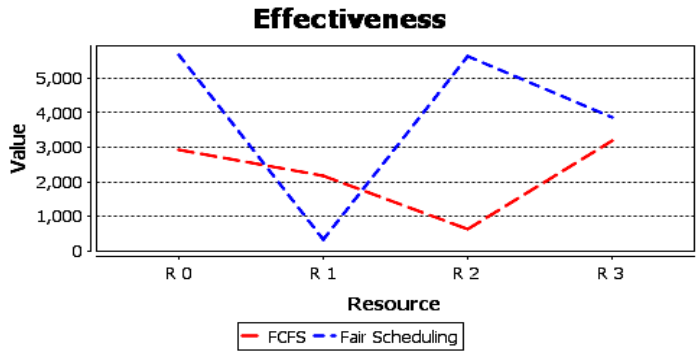
---Total No of jobs: 4

Job ID: 0=0
Job ID: 1=1
Job ID: 2 =2
Job ID: 3 =3

---Resource creation

Initializing GridSim package
Initializing
Created Res0 with id = 5
Creating Res1 with id = 9
Creating Res2 with id = 13
Creating Res3 with id = 17
---Resource status
Fx00=4
W0=1416
Value function of site 0 is 5664
Fx11=4
W1=84
Value function of site 1 is 336

Fx22=4
W2=1404
Value function of site 2 is 5616
Fx33=4
W3=964
Value function of site 3 is 3856
Minimum loaded site is 1
That is 336



Processor capacity: 40
Job id
0
1
2
3
Demand Rates
8.0
107.0
6.0
90.0
Weights
2
1
1
2
First Iteration
8.0
6.666666666666667
6.0
13. 3333333333334
Second Iteration
8.0
8. 666666666666668
6.0
17.333333333333336

Job allocation:
Gridlet1 is submitted to site Res1 and ID is: 9
Gridlet3 is submitted to site Res5 and ID is: 25
Gridlet0 is submitted to site Res2 and ID is: 13
Gridlet2 is submitted to site Res3 and ID is: 17

Initializing…
Starting GridSim version 5.0
Entities started.
Sim_system: No more future events
Gathering simulation data
Grid Information Service: Notify all GridSim entities.
Simulation completed.

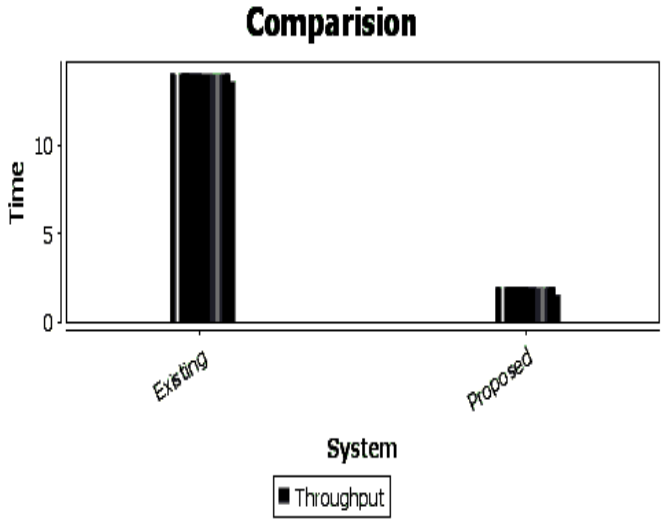
Gridlet ID	Status	Output Size	Cost
0	Success	300	0.0
1	Success	300	0.0
2	Failure	300	0.0
3	Success	300	0.0

Fault Manager sends the failure message details to user
Failure site Gridlet id 2 Status FAILURE
Assign job 2 to Gridlet with name = gridlet_4 and ID =4
Initializing Gridsim package
Initializing…
Created Res4 with id = 5
Gridlet 4 is submitted to site Res 4

Gridlet Number	Status
4	Success

------Job Execution

Gridlet Number	Status
0	Success
1	Success
2	Failure
3	Success
4	Success (Recovery Done)



In Fig. 4. Comparison of FCFS and Min-Max Algorithm

In Fig.4**,** we compare the throughput of FCFS and Min-Max algorithm so proposed algorithm give better response time in within fraction of seconds, all jobs executed in fraction of seconds so here maximum resource utilization done and also increase the performance of grid environment so performance of grid is improved.

V. Future Work

In this paper, we presented a highly de-centralized, distributed and scalable algorithm for scheduling tasks and load balancing resources in Grid Environment. Our algorithm providing an effective scheduling and resource management strategy. We introduce a new Min-Max Fair scheduling algorithm based on minimum efficiency of site. So that gives better performance. Still we can increase the performance of Grid, in future work

we can use efficient fault tolerant mechanism that gives better results.

Conclusion

Load balancing is important issue in the grid computing environments. We propose a new way to load balancing. The Min-Max fair scheduling algorithm is played very important role in the job dispatch phase. Using this Min-Max algorithm the jobs are scheduled. The fair rates are very important in the fair scheduling algorithm and another issue in grid is Fault Tolerant. The fault is tolerated by execute the job in another resource. By this jobs are successfully executed. Here we achieve high throughput and resource utilization is done so that performance of Grid Environment is increased.

REFERENCES

[1] Nikolaos D, Doulamis, Anastsios D. Doulamis, Emmanouel A. Varvarigos and Theodora A. Varvarigou, "Fair scheduling algorithm in grids", IEEE transactions on parallel and distributed systems, Vol. 18, No.11, November 2007, pp. 1630-1648.

[2] Buyya, R., Abramson, J., and Giddy, j/ Nimrod/G: architecture for a resource management and scheduling system in a global computational Grid. 4th IEEE conf. on High-performance computing, 2000.

[3] Kai lu and Albert Y.Zomaya "A Hybrid policy for Job Scheduling and Load Balancing in Heterogeneous Computational Grids" 6th International Symposium on parallel and Distributed Computing, IEEE, 2007.

[4] Manish Arora, Sajal K. Das and Rupak Biswas "A De-centralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments", Proceeding of the International Conference on Parallel Processing Workshops, IEEE, 2002, pp. 1-7.

[5] Yajun Li, Yuhang Yang and Rongbo Zhu "A Hybrid Load balancing Strategy of Sequential Tasks for Computational Grids", International Conference on Networking and Digital Society, IEEE, 2009, pp. 112-117.

[6] S.K. Karthik Kumar, M. Udhaya Preethi and P. Chitra "Fair Scheduling Approach for Load Balancing and Fault Tolerant in grid environment", IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology, 2013. pp. 446-451.

[7] Jing Wei-peng, liu Ya-qiu and Wu Qu "Fault-tolerant task scheduling in Multiprocessor systems based on Primary-Backup Scheme", IEEE, 2010, pp. 670-675.

[8] Jasma balasangameshwara, Nedunchezhian Raju "A Hybrid policy for fault tolerant load balancing in grid computing environment", Journal of network and Computer Applications, Vol. 35, Issue 1, January 2012.

[9] Hwang s, Kesselman C,"A flexible framework for fault tolerance in the grid", Journal of grid computing, 2003.

[10] N. Budhiraja et al., S. Mulender "The primary Backup Approach", Distributed systems, pp.169-197, 1993.

[11] Li K, "Optimal load distribution in non dedicated heterogeneous cluster and grid computing environment", Journal of Systems Architecture: The EUROMICRO Journal 2008.

[12] Nandagopal Malarvizhi, Rhymend Uthariaraj V, "Decentralized dynamic load balancing for multi cluster grid environment" Future Generation Computer Systems 2009.

[13] L. Anand, D. Dhose, V. Mani, "ELISA: An estimated load information scheduling algorithm for distributed computing systems", Computers & Mathematics with applications, Vol. 37, Issue 8, April 1999.

Authors Profile

Mr. Mahesh Reddy G pursed Master of Technology from Vignan's University, India in year 2014. He is currently working as Assistant Professor in Department of Computer Science and Engineering. His main research work focuses on Grid Computing and IoT. He has 2 years of teaching experience and 2 years of Indistrial Experience.

Mr. Lakshmi Narayana G pursed Bachelor of Science and Master of Science from JNTUK, India in year 2012. He is currently working as Assistant Professor in Computer Science and Engineering. His main research work focuses on Grid Computing, Big Data and IoT. He has 6 years of teaching experience .

Mr. Srinivasa Reddy K pursed Master of Technlogy from JNTUK, India in year 2016. He is currently working as Assistant Professor in Computer Science and Engineering. His main research work focuses on Grid Computing, DWDM and IoT. He has 2 years of teaching experience.

