SCHEDULING IN VIRTUALIZED GRID ENVIRONMENT USING HYBRID APPROACH

S.Thamarai Selvi, Ponsy R.K.SathiaBhama, S.Architha, T.Kaarunya, K.Vinothini

Department of Information Technology, MIT, Anna University.
Email: ponsy_rk@yahoo.co.in

ABSTRACT

A typical grid application needs requirements in terms of memory, software and so on. Scheduling those applications and giving the results back to the client is the prime goal. In this paper, the scheduling algorithm is proposed for assigning jobs to the physical resources. When the software requirements of the application are not satisfied by the physical resource, the jobs are scheduled to a virtual machine. This virtual machine is created such that it satisfies the requirements of the job. This is the hybrid approach of scheduling proposed in this paper. The algorithm allocates dynamic priority to the jobs based on execution time and deadline and classifies them into multi-queues based on priority allocated. The bipartite graph is used for resource matching which gives the best resource match and reduces the conflicts between jobs for a particular resource. The approach in this paper is different as the dynamic user defined virtual environment is created only when there is no such physical environment already available. The reliability of the system is improved by proactive failure detection and RPC fault tolerance mechanism. The proposed system handles four categories of faults. Load balancing is also achieved through receiver initiated strategy.

KEYWORDS:

Dynamic priority, Bipartite graph, Virtualization, Proactive detection, Receiver initiated strategy

1. INTRODUCTION

Grid computing environment is a distributed, shared environment implemented via the deployment of a persistent, service infrastructure that supports the creation and resource sharing within distributed communities. Resources can be computers, and data, all connected through the internet and a middleware software layer that provides basic services for security, monitoring, resource management, and so forth. Resources owned by various administrative organizations are shared under locally defined policies that specify problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.

Grid computing virtualizes distributed computing and data resources such as processing, network bandwidth and storage capacity to create a single system image, granting users and applications seamless access to vast IT capabilities. The goal is to create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of heterogeneous systems sharing various combinations of resources.

In such an environment, it is necessary to schedule the jobs to the appropriate resources that exactly match the requirements of the job. And also it is essential to decide the job that should be given a higher priority so that it can be scheduled first. This priority is given based on execution time and deadline. The factor, execution time is considered to balance the load in the resources in

DOI : 10.5121/ijgca.2010.1101

each cluster. And the deadline factor is used to determine the emergent jobs that have to be scheduled first.

In this paper, the proposed algorithm schedules both the resources and jobs. Scheduling of jobs to the resources with the parameters like processing time, deadline and waiting time is the job scheduling. Selection of resources for allocating the jobs, fitting the requirements of an application in an intelligent way with the parameters like, memory, latency, RAM size, CPU speed, bandwidth, software requirements is resource scheduling.

The resource to be assigned to the job is decided with the help of bipartite graph. In the mathematical field of graph theory, bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V; U and V are independent sets. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles. Here the U and V disjoint sets are jobs and the set of resources matching their requirements. The exact resource is decided with the help of the weight on the link between U and V. The link which has the lowest weight represents the perfect resource for the job. Scheduling the jobs to the physical resources reduces the overhead of creating virtual machines. But at the same time, it is required to create the user defined virtual environment to schedule their jobs when there is an absence of a physical machine satisfying the requirements of the job.

2. RELATED WORKS:

The job scheduling algorithm proposed in grid environment in [7] is as an optimization problem. The result is a cost model for improving the grid environment in term of make-span time and grid efficiency. The algorithm is developed based on an ant colony optimization which finds the suitable resources and allocates them appropriately, and hence they can process each job with efficiency. The algorithm is able to find an optimal processor for each machine to be allocated to a job. However, the result of ACO algorithm is still not good enough when compared with MCT and Min-Min. This paper proposes a resource selector and scheduler algorithms that produce even better results as it focuses on “all jobs scheduled and no jobs suffer”.

The workload of a Grid enabled cluster is analyzed in [13] and it proposed an infinite Markov based model that describes the process of jobs arrival. The proposed scheduler uses even better queuing model which takes care of the task arrival interval as well as queue capacity that can be processed by the scheduler at an instance.

In [9], new algorithms are proposed based on software redundancy to tolerate permanent and timing failures. They have considered heterogeneous real time distributed systems with point-to-point communication links for this research work. These proposed algorithms are resigned to meet two basic objectives (i) the execution of backup versions should not hinder the execution of the primary version of the tasks, and (ii) when the primary task fails to meet its deadline the backup instance should then be executed but it should be executed from the point of last correct sub task executed by the primary version. This paper proposes fault tolerance through RPC and replication. Initially the category of fault is identified and then appropriate technique is invoked. Using RPC prevents multiple versions of output from reaching the client.

The Genetic algorithm (GA) implementation coupled with a flexible simulation system to support the job scheduling on computational grids is proposed in [12]. The results obtained from their GA implementation are very promising; they have identified a setting of parameters and operators that outperform the current best GA Job Scheduling system on a static scheduling benchmark.
The **parameters** considered were
(i) A *number of independent (user/application) tasks* that must be scheduled

(ii) A *number of heterogeneous machines*
(iii) The *workload* of each task.
(iv) The computing capacity of each machine
(v) The expected time to compute
The **GA operations** considered were
(i) Select Random: this chooses the individuals for the pool of pairs to cross uniformly at random.
(ii) Select Best: this chooses only individuals with better fitness.
(iii) Binary and N-Tournament Selection
The proposed algorithm considers all the above said parameters and includes the job transfer time too. This proposed algorithm apart from scheduling also ensures the job's successful execution by improving the reliability of the system.

In [1] the system that uses virtual machines as the building blocks to construct execution environments that span multiple computing sites is proposed. An execution environment is a network of virtual machines created to fulfill the requirements of an application, thus running isolated from other execution environments. These environments can be extended to operate on Cloud infrastructure, such as Amazon EC2. The paper proposed is a hybrid environment where the jobs are scheduled on to physical machine and when the requirement of the job is not satisfied by a physical machine, a dynamic user defined virtual environment is created and jobs are scheduled. This reduces the overhead in creating virtual machines when there is availability of physical machine satisfying the job.

3. ARCHITECTURE

User submits the job to the broker which is comprised of 4 managers, scheduler manager, information manager, transfer manager and execution manager. This paper focuses on scheduler manager. Scheduler manager has scheduling algorithms to produce efficient schedule for the jobs in the job queue. Jobs in the job queue are ordered according to priority based on execution times and deadline.
This queue is fed to the resource set selection algorithm where the resources matching the software requirements are chosen and best fit is made according to memory requirements and matched with the help of bipartite graph. This matching information is taken by the transfer manager that uses GridFTP to transfer job to the respective resource and then the execution manager causes the execution of the job in that resource. When the software requirement of the job doesn't match ,virtualization comes into play where virtual machines are created with the requirements of the user and the job is scheduled to the virtual machine.

User submits the job to Job Submission Queue by filling the jsp form. The server collects the job information in the socket port and puts into the xml file. It also starts the thread for each client which will be alive till the job is scheduled. Priority based Job Selector collects the jobs present at that instance and adds them into the linked list based on the priority. This priority assigned to the jobs based on factors execution time and deadline. Resource Matcher, takes each job from the linked list in the order of priority and a set of resource matching that particular job is selected. This matching is done with the factors software and memory.

From the set of resources selected for the particular job, the best resource is fixed using bipartite graph weight calculation. Factors for weight calculation are network delay, bandwidth, queue waiting time and number of resources.

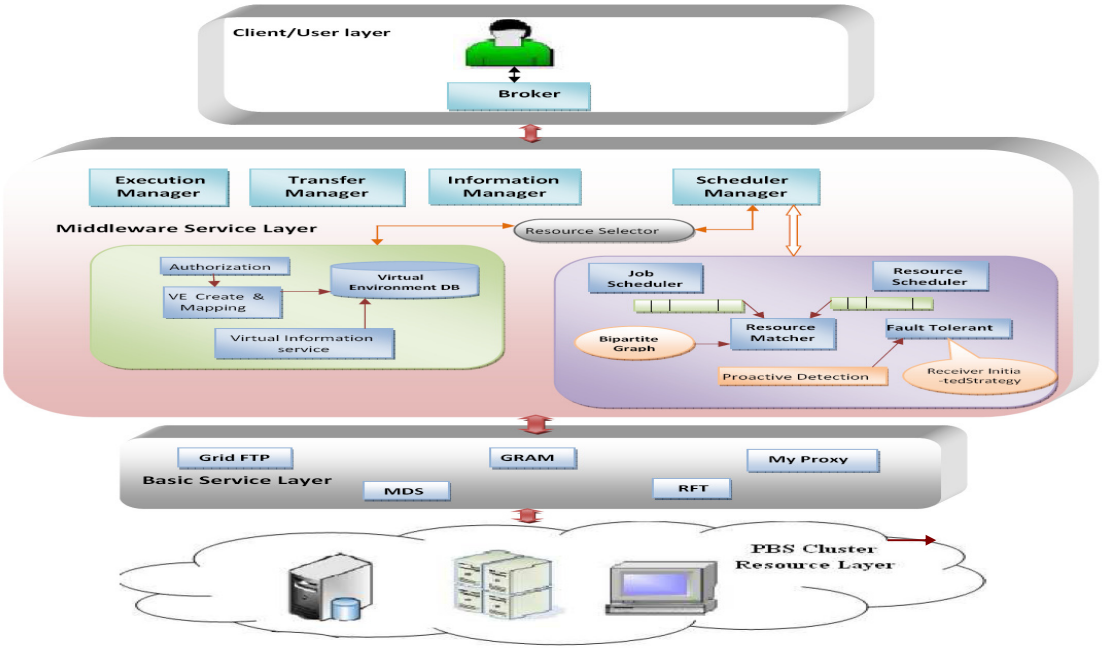

Figure1.Priority scheduling architecture

If the job in the linked list is not scheduled due to non availability of software requirements, a resource which has the capability to create the virtual machine with the required memory, software, CPU speed and RAM size is given a request to create the dynamic environment required by the job.

If the job in linked list is not scheduled due to non availability of memory allow the job to wait in the list and schedule it after an interval of 2min. If the job is not scheduled again repeat it such that the interval between submission and rejection is not more than 6min.

After the submission of job is done, it is monitored for four categories of failure and RPC fault tolerant technique is implemented to improve the reliability. The load is too balanced by implementing the receiver initiated strategy.

4. SCHEDULING ALGORITHMS

4.1. Dynamic Priority allocation

The jobs are allocated priority dynamically based on execution time and deadline and fed into multiple queues. Average of the execution times of the jobs at that instant is taken as a threshold and same is done for deadline. Priority is given in the order as in the table[1]

Deadline	*Execution Time*	*Priority Queue*
<=	>	1
<=	<=	2
>	>	2
>	<=	3

Table. 1 priority order

Within the queue, the jobs are put in the descending order of the ratio,

Execution time/Deadline

Job scheduling is repeated for every set of jobs taken at regular intervals. The newly arrived jobs are placed at the appropriate queue and the jobs in the interval of changed threshold are alone rescheduled to the perfect queue. In this way, the jobs are prioritized dynamically at regular intervals which prevent the starvation as the priority of the job varies with the varying threshold at regular intervals. The algorithm for allocation of priority is shown in figure[2].

```
Algorithm1:
Begin
Get Execution time and   Deadline for the job
Compute the threshold value for these inputs
 for each job
  if  job_deadline <= threshold_deadline   &&
     job_executiontime>=threshold_executiontime
   then insert that job into queue1
     elseif  job_deadline<=threshold_deadline &&
           job_executiontime<=threshold_execution
          time|| job_deadline>= threshold_execution
          time|| deadline && job_executiontime>=
          threshold_ executiontime
           then insert the job into queue2
             elseif  job_deadline>=threshold_dead
                   line &&  job_executiontime<=
                  threshold _executiontime
                  then insert the job into queue3
                  endif
           endif
endif
Compute the execution time/deadline ratio for each job
then arrange the queue  in decreasing order
    end
end
```

Figure2. Min-wait deadline first algorithm

In shortest job first algorithm, it keeps the no of jobs waiting for the processor as low as possible. The main criterion for real time operating systems is deadline, which is met by earliest deadline first algorithm. But the jobs' waiting time is not taken, which might be the reason for the late response. The proposed scheduling algorithm minimizes the waiting time in the scheduler queue as well as in the job queue and also meets the deadline of the jobs thus proving to be better than the standard algorithms available.

4.2. Resource Set Selection

The jobs from the queues are taken one by one. The resource information is gathered by querying MDS (Monitoring and Discovering Service) with the software and memory requirements of the job. To achieve the best fit resource scheduling, those resources are selected whose memory availability is almost equal to that of the job's requirement. For this purpose, a threshold value is

fixed such that the memory availability of the resource must be equal or greater than job's requirement but less than the threshold level. Thus a resource set is selected for each job in the queue.

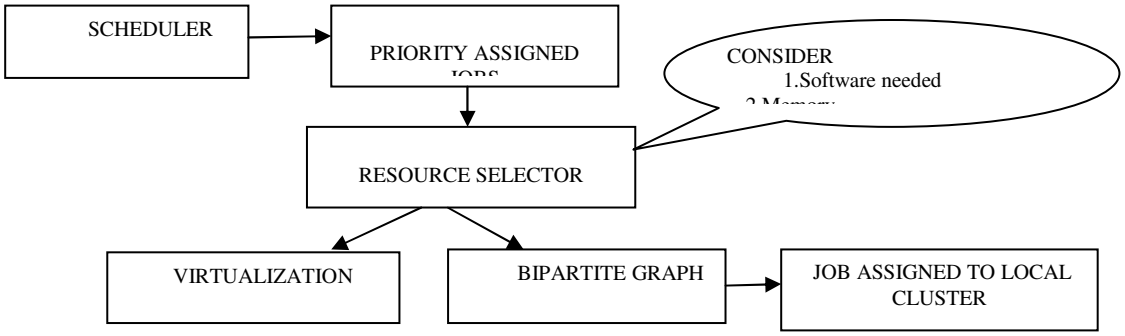

Figure3. Resource selection process

```
Algorithm2:
Begin
Initialize extendvalue=0 and value=10
Compute the job queue
  foreach job
    Get the resource information from MDS
    if job_software==resource_software
     then insert the resource node into the
          resource queue
     Calculate value=resource_memory+value+
                        Extendvalue
          if job_memory>=resource_memory &&
             resource_memory<=extendvalue
          then reject the other resources in the
               resource queue
             Call bipartitegraph();
          Else calculate value again
          endif
      else Call Virtualization();
      endif
    end
end
```

Figure4. Physical vs Virtual Resource selection algorithm

4.3. Resource Matching using bipartite graph

From the resource set selection algorithm, there is a set of resources for each job. The bipartite graph is constructed with two disjoint sets – job and resources based on the output of previous step. Weight is calculated for each link in the bipartite graph based on the parameters such as network delay, queue length, performance of the resource.

$$\mathbf{M(u_i)=F(min(A_{ij}),n_i)}$$

where

A_{ij}= round(JTT_i, OWT_j) where

JTT_i = Job transfer time that takes into account the delay in the network and

$bandwidth_j$ = bandwidth

OWT_j = waiting time in the queue of the $resource_j$

n_i = number of resources requested by the job_i

Link which has the lowest weight is chosen and the job is scheduled to the corresponding resource in the selected link. If two links in the bipartite graph constructed have same weight then the one which has minimum memory is allocated to the job for best fit resource scheduling. If the memory availability of the resources is also the same, then the bandwidth factor is considered for allocation. The above steps are repeated for each job.

The **advantage of using the bipartite graph** is

1. giving the best matched resource for the job
2. preventing the conflict for the resources between the jobs

The algorithm for resource matching is as follows.

```
Algorithm2:
Begin
Compute the job queue and resource queue
  for each job from the job queue
     Construct the link between job and the
      resources from the resource queue
     Compute the link weight with network delay
        and waiting time
       Find min (linkweigtht) using the
            M(ui)=F(min(Aij),ni)
     If more than one min link weight
       then Select the best fit resource
    endif
end
```

Figure5. Bipartite Resource matching algorithm

4.4. Virtualization

Virtualization is a technique for hiding the physical characteristics of computing resources to simplify the way in which other systems, applications, or end users interact with those resources.

Virtualization lets a single physical resource (such as a server, an operating system, an application, or storage device) appear as multiple logical resources; or making multiple physical resources (such as storage devices or servers) appear as a single logical resource. The script for gathering the information about the software available in the resources is written in the mds parser. In the linux operating system, this information can be obtained from /etc/profile.

If no existing resource has the software required for the job, virtualization concept is used. The basic two steps for creating the virtual machine is,

(i)Identify the Cluster Head Node:

Identification of the resource which has the memory required for creating VM for the job and also the RAM of the resource is considered. The job transfer time is another factor considered for selecting the resource for the job.

(ii)Creation of VM:

Once the resource is selected, VM software is invoked for VM creation and required softwares are installed and the job is transferred to the resource for execution.

4.5. Rescheduling

When the memory availability is not satisfied, the job is made to wait in the queue and the timestamp is noted. The rescheduling interval is fixed as 2mins. The resource availability is checked after 2mins, if such a resource is found, the job is allocated to the resource for execution. This rescheduling is done until the timestamp is not exceeding 6mins (ie) the time from the instance at which the job is submitted.

4.6. Fault Tolerance

This paper categorizes the faults as
1. **Code errors** – from the error messages returned by the GRAM, if any exceptions occur in the job code, the error messages are given to the user for rectification of the code.
2. **Software Unavailability** – if the requires software for the job becomes unavailable in the machine; VM is created in that machine if it satisfies the VM requirements. If the machine doesn't have such a capability, the job is rescheduled
3. **Network failure** – This type of fault is identified by the 'destination unreachable' message.
4. During execution of the job in a resource, the gram service stops
When the fault 3) and 4) occurs, our fault tolerance mechanism is introduced.

4.6.1. Receiver initiated strategy:

Once the job is submitted to a resource; resources which are lightly loaded, initiates themselves with their information to the scheduler that they are ready to accept the job. These resources are verified for its capability to take up the job such as memory, software availability and deadline constraints of the job. This saves the time in search for the resources suitable for the job if the allocated resource for the job fails. Also supports balancing the load among the nodes.

4.6.2. RPC fault tolerance:

If the failure of the resource occurs, the least loaded resource is loaded with the backup copy of the job and the primary resource gives a RPC to the backup copy thus invoking the execution of the job. Thus the replication of jobs when there is zero probability of failure is prevented minimizing the wastage of resource. RPC is used to prevent multiple occurrences of result as the result of the call is returned to the client by the primary caller. If the primary caller fails, its backup incarnation assumes the role of the primary and replies to the client.

5. PERFORMANCE ANALYSIS:

The proposed model has increased the probability of scheduling the submitted jobs. This can be proved by analyzing the model. This initially schedules jobs onto the physical matching resource. Rescheduling mechanism is provided if such a resource is available but in busy state. If the required resource is not found, the scheduler initiates the creation of virtual machine onto the resource that has the capability and submits the job to it. Thus jobs are scheduled at the higher probability. This model also deals with the successful execution of the submitted jobs. This can be shown by the reliability model proposed below.

Here two models are considered to show the reliability of our system.

1. Job that requires one node
2. Job that requires more than one node

When only one node is required for execution of the job, the reliability of the system with fault tolerance is evaluated by exponential distribution. When the node has more probability of failure, job is submitted to another node through RPC. Considering m as the number of nodes that is available for tolerating the failure of that particular job, reliability of the system is given by

$$\mathbf{R(t) = 1 - \prod_{i=1}^{m} q_i(t)}$$

$$\mathbf{= 1 - \prod_{i=1}^{m} \{1 - p_i(t)\}}$$

Where

$\mathbf{p_i(t)}$ – probability that the system is good

$\mathbf{q_i(t)}$ – probability that the system will fail

$$\mathbf{p(t) = \exp(-\lambda t)}$$

The m nodes selected is based on the following criteria

1. Set of lightly loaded nodes that initiated itself
2. Should match the requirements such as memory and software
3. Deadline of the job > current time + job transfer time + execution time of the job

Mean time to failure MTTF for such a system is given by,

$$\mathbf{MTTF = \int_0^{\infty} [\, 1 - \Pi_{i=1}^{m} (1 - \exp(-\lambda_i t))]\, dt}$$

If a job requires more than one node for execution the reliability model is based on binomial distribution. Taking m as the number of available nodes, the probabilities that x nodes are successful out of m nodes is given by

$$\mathbf{P(m,x) = B(m,x)p^x(1-p)^{m-x}}$$

$$\mathbf{B(m,x) = \text{binomial coefficient}}$$

If k nodes are required for execution of job, then system reliability is the sum of binomial probabilities x varying from k to m.

$$\mathbf{R = \sum_{i=k}^{m} B(m,i)\, p^i(1-p)^{m-i}}$$

The mean life of the system is

$$\mathbf{MTTF = \int_0^{\infty} R(t)\, dt}$$

where R(t) is the reliability function for the a system which executes job with the requirement of more than one node.

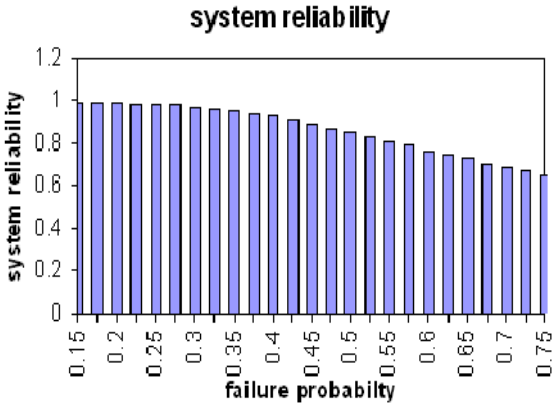

Figure6. System reliability vs failure probability

From figure 6, it is shown that the reliability of the system in executing the jobs successfully is higher even when the failure probability of the each node in the system is more. The above graph depicts a 3 unit system. From observations, it is found that in grid environment, the resource availability for various applications is still higher than 'm' assumed here.
From figure 7, any increase in resource availability would obviously increase the reliability of the system and hence decrease the failure of the scheduling system.

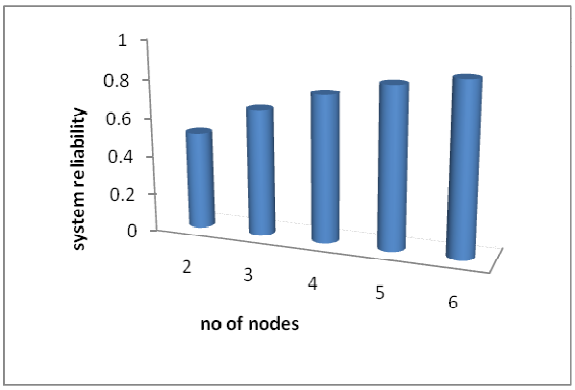

Figure7.system reliability vs no of nodes

The figure 8 gives a comparison between the theoretical and implemented reliability values for the proposed model. When the proposed model is implemented, it was observed that the submitted jobs were scheduled, thereby reducing the rejection probability and the scheduled jobs were successfully executed due to the reliable system developed.

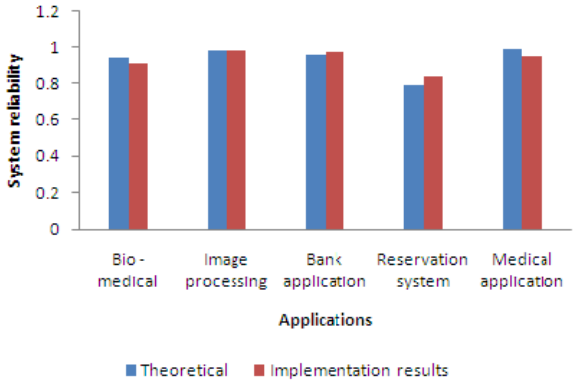

Figure8. System Reliability for proposed system

6. CONCLUSION

Thus the job and resource scheduling combined for efficient execution of jobs submitted by the clients. This algorithm combines the advantages of standard algorithms such as shortest processing time, longest processing time, and earliest deadline first. To address the problem of resource unavailability and software mismatch, virtualization technique is required. This paper currently investigates how to extend the work to include virtualization and how the scheduling approach presented in this paper can be applied to physical machine as well as virtual machine.

With the help of virtualization a dynamic user defined environment is created, so there is less possibility of rejecting a job due to unavailability of resource. Since this is the hybrid of virtual and physical machine scheduling, overhead in creating virtual machine in the presence of physical resource is reduced.

This paper gives more importance for scheduling the jobs to the resource and certain types of faults are identified and handled using proactive detection and RPC fault tolerance mechanism. This could further be extended to handle some more faults to increase the reliability of the scheduling system proposed.

REFERENCES

[1] Alexandre di Costanzo, Marcos Dias de Assunção, and Rajkumar Buyya, (2009) "Building a Virtualized Distributed Computing Infrastructure by Harnessing Grid and Cloud Technologies", Grid Computing and Distributed Systems Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia.

[2] Su-Hui Chiang; Vasupongayya, S, (2008) "Design and Potential Performance of Goal-Oriented Job Scheduling Policies for Parallel Computer Workloads Parallel and Distributed Systems", IEEE Transactions on Volume 19, Issue 12, Page(s):1642 – 1656.

[3] Yanbin Liu, S. Masoud Sadjadi, Liana Fong, Ivan Rodero, David Villegas, Selim Kalayci,NormanBobroff, Juan Carlos Martinez, (2008) "Enabling Autonomic Meta-Scheduling in Grid Environments" ,IEEE International Conference on Automonic Computing.

[4] Thamarai Selvi Somasundaram, Balachandar R. Amarnath,Balakrishnan. P, Kumar R, Rajena r K, Rajiv R,Rajesh Britto.G,Mahendran. E, Ma d usudhanan. B, (2008) "Achieving Co-allocation through Virtualization in Grid environment ".

[5] A.J. Sanches Santiago, A.J. Yuste, J.E. Munoz Exposito, S. Garcia Galan, J.M. Maqueira Marin, S. Bruque, (2008) "A dynamic-balanced scheduler for Genetic Algorithms for Grid" Computing Wseas Transactions on Computer.

[6] Subodha Kumar Kaushik Dutta Vijay Mookerjee (2007) "Resource Scheduling in Grid Computing Networks to Maximize Business Value", http://ssrn.com.

[7] Siriluck Lorpunmanee, Mohd Noor Md Sap, Abdul Hanan Abdullah, Aboamama Atahar Ahmed, (2007) "Multi constraint dynamic scheduling of independent jobs onto grid environment", Jumal Teknologi Maklumat, Jilid 19, Bit. 1.

[8] Gaurav Khanna , Umit Catalyurek , Tahsin Kurc , Rajkumar Kettimuthu , (2007) **"**A Dynamic Scheduling Approach for Coordinated Wide-Area Data Transfers using Grid FTP"National Science Foundation under Grants #CCF-0342615.

[9] Aser Avinash Ekka, (2007) "Fault Tolerant Real Time Dynamic Scheduling Algorithm For Heterogeneous Distributed System", NIT Rourkela.

[10] Swapnil Srikanth Bagul,Dheepak R.A,Nilesh Ranade,Shubhashis sengupta, (2006) "Dynamic Integra tion of Heterogines Enterprise data-a grid based Approach" infosys Technologies limited.

[11] Tao Zhu, Yongwei Wu, Guangwen Yang, (2006) "Scheduling Divisible Loads in the Dynamic Heterogeneous Grid Environment", INFOSCALE '06, Proceedings of the First International Conference on Scalable Information Systems.

[12] Javier Carretero, Fatos Xhafa, (2006) "Use of genetic algorithms for scheduling jobs in large scale grid applications", Technological and economical development of economy, Vol XII, No 1, 11–17.

[13] Emmanuel Medernach, (2005) "Workload analysis of a cluster in a Grid environment", Workload characterization and modeling.

[14] Paulruth xuxian jiang,Dongyan Xu,Sebestin goasguen, (2005) "Virtualization for Dynamic Computational Domains"Proceedings of IEEE International Conference for Cluster Computing.

[15] Marai Chtepen, Prof.Dr.ir.Bart Dhoedt, Prof.Dr.ir.Peter nrolleghem, (2005) **"**Dynamic Scheduling in Grid System" Sixth firW phD Symposium,Ghent University.

[16] Ramin Yahyapour, Philipp Wieder**,** (2003) "Grid Scheduling**"** Grid Scheduling Architecture Research Group (GSA-RG).

[17] Ian Foster ,Carl Kesselman,Steven Tuecke, (2001) "The Anatomy of the Grid – Enabling Scalable Virtual Organization" The Internet and Beyond. National Academy Press.

[18] Norman Bobroff, Liana Fong, Selim Kalayci, Yanbin Liu, Juan Carlos Martinez, Ivan Rodero, S. Masoud Sadjadi, and David Villegas, "Enabling Interoperability among Meta-Schedulers" Eighth IEEE International Symposium on Cluster Computing and the Grid.

[19] F.Berman, S.Figueria, J.Schopf ,G.Shao and R.Wolski, (1996) "Application-Level Scheduling on Distributed Heterogeneous Network"s, in Proc. Of the 1996 ACM/IEEE Conference on Supercomputing, Article No: 39, Pittsburgh, Pennsylvania USA.

[20] Gilbert C. Sih,, and Edward A. Lee, (1993) "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures" IEEE transactions on par- allel and distributed systems. VOL. 4. NO 2.