A New Architecture for Autonomous Grid

Saket Kumar Mishra
B.Tech
Computer Science & Engineering Department,
Sikkim Manipal Institute of Technology, Majhitar, Sikkim

Shekhar Pratap Sinha
B.Tech
Computer Science & Engineering Department,
Sikkim Manipal Institute of Technology, Majhitar, Sikkim

Chinmoy Kar
Assistant Professor
Computer Science & Engineering Department,
Sikkim Manipal Institute of Technology, Majhitar, Sikkim

ABSTRACT
Autonomous systems are inspired by biologically systems which their goal is to manage themselves with minimal involvement of managers. Autonomous is a concept that brings together many fields of computing with the purpose of creating any systems that self-manage. Autonomous strives to make future systems even more receptive, resilient and responsive. Autonomous is suitable for a computational grid because of its dynamic and autonomous nature. Environment of computational grid is inherently large, complex, heterogeneous and dynamic and its state changes over time, so continues monitoring by using Autonomous putting more of the burden on the computers and less on the system administrators. This paper proposed an architecture which explain how Autonomous works on gird environment and algorithms which increase reliability computational grid.

General Terms
Autonomous Grid

Keywords
Autonomous system, Grid scheduling

1. INTRODUCTION
T he concept of grid computing is gaining popularity as it is all about using the unused resources connected to each other through the network from anywhere in the world. Irrespective of the location of the resources, it could be used to perform many large computations without having to bother about the availability of the resources. Thousands of heterogeneous resources can leave or join the Grid at any point of time. Therefore there is a need to manage and monitor [1] these resources continuously in order to avoid any malfunctioning of the resources which may affect the performance and the efficiency of the grid [2]. Monitoring, adding and removing resources in grid manually are not possible as the complexity of the grid is increasing day by day. There is always the possibility of security risk and resource failure [3] due to the dynamic nature of the Grid. Hence the grid can be managed more effectively using the dynamic algorithms which can make decision based on the current situation.

To handle the above mentioned problems, there is a need to introduce autonomic behavior in the grid which can deal with the issues at the lower level and frees the administrator to handle the issues at higher level [4]. Autonomic computing is inspired from human autonomic system and is all about making a system autonomic by implementing the autonomic computing features in it. In this project a novel architecture has been proposed which uses the Autonomic manager to introduce the autonomic features in the grid which manages and monitors the entire grid. The Autonomic Manager [5] also protects the grid from any malfunctioning of any resource while ensuring the utmost performance of the grid.

2. AUTONOMIC COMPUTING
Autonomic computing is a system which can self-configure, self-optimize, self-protection and self-heal itself with minimum human intervention. The term "autonomic" is coined by the IBM and is a biological term. It is inspired by the autonomic nervous system that takes care of the actions such as respiration; digestion etc., without any external assistance. It is not possible for us to take decision for every action that our body requires to take. This term was proposed by the International Business Machine (IBM) based on the human autonomic nervous system which adapts itself without needing any attention [6]. Therefore, Autonomic computing is a self-managing system that can manage itself with minimal intervention or assistance from outside.

2.1 Architecture of Autonomic Computing
IBM suggested architecture for autonomic computing which has several basic components. These basic components work together to achieve the autonomic behavior in any system. These components make the decision based on the current conditions and adapt to the conditions accordingly to changes with the dynamically changing environment. IBM has given several components [6] to achieve autonomic behavior in a system. These components include manual managers, autonomic managers, touch points and managed resources. These components share the common knowledgebase and takes decision accordingly.

2.2 Features of Autonomic computing
There are four basic features of the Autonomic Computing that has been proposed by the IBM in 2001. Any system which incorporates all these four features becomes a self-managing system or in other words, to make a system autonomic, these four features must be implemented on that system. The following are the four features of the autonomic computing:

2.2.1 Self-Configuration
This feature of autonomic computing deals with configuring the resources to obtain the maximum performance. This also deals with adapting with the changing environment with minimal assistance from outside. Dynamically adding a resource to a grid or removing a resource from a grid is one of the examples of the self-configuration feature.

2.2.2 Self-Optimization
This feature of autonomic computing deals with using the best resource from all the available resources. There may be more than one available resource for performing one or more job. Therefore, this optimizes the resources in the best possible way using the algorithms that make decisions that is based on the current scenario of the system and its resources.

2.2.3 Self-Protection

This feature of autonomic computing deals with protecting the system without requiring the attention of the administrators. This protects the system against any external intrusion which may cause the instability of the system.

2.2.4 Self-Healing

This feature of the Autonomic computing deals with healing the infected system and protects the system from any further damage. It isolates the infected resource or system and then attempts to repair it. Once the resource is repaired, it is again added to the system.

3. GRID COMPUTING

Grid consists of large number of heterogeneous resources working in a dynamic environment and utilizes the resources to the best of its capacity. Due to large number of resources connected to each other, the grid is becoming very complex. Any resource can join the grid anytime and also it can leave the grid whenever it wants. Grid computing gathers all the computing power of all the unutilized resource at one platform named Grid to provide the computing power wherever needed in the world. Grid removes the need to buy additional resources to meet the demand of the customers which is temporary. The additional resource can be acquired temporarily from the grid to meet the demand which will be more economic then buying separate resources for using it for short period of time.

4. A NEW MODEL FOR AUTONOMOUS GRID

4.1 Work flow of proposed model:

The following steps explain the proposed grid architecture. The following steps numbers are also mention in figure 2.

1. Client will submit the job to the 'Job Queue'
   1.1. 'Job Receiver' will receive the job if any, from other grid and will add it to the 'Job Queue'.
2. 'Self-Optimization Manager' will check the 'Temp Job Queue for the Job.
3. 'Self-Optimization Manager' will check the 'Job Queue, if no jobs are there in the 'Temp Job Queue'.
4. 'Self-Optimization Manager' will check for the resources in the 'Passive Resource'. If there are no resources in the 'Passive Resource', it will then check 'Active Resource'.
5. If there is no resource in either 'Active Resource' or 'Passive Resource', then 'Self Optimizer' will submit the job to 'Job Transmitter' in 'Job Exchange Manager'.
6. If 'Self Optimization Manager' finds the resource in the 'Passive Resource', it will schedule the job and call the 'Job Dispatcher'. If 'Self Optimization Manager' finds the resource in 'Active Resource' then it will wait for the resource to be transferred to the passive resource.
7. 'Job Dispatcher' will dispatch the job to the respective resources.
8. 'Monitoring Manager' will start creating checkpoints for each resource executing the job and will wait for an acknowledgement from the resource either about the successful completion of the job or an incomplete execution of the job.
   8.1. 'Self-Protection Manager' will start checking for any malfunction in the resource while executing the job.
9. If 'Self-Monitoring Manager' receives an acknowledgment about incomplete execution of the job,
10. then it will add the partially executed job from the latest checkpoint created by the 'Monitoring Manager' to the 'Temp Job Queue'.
11. 'Monitoring Manager' will call the 'Self Configuration Manager' to remove the resource from the 'Active Resource' and add it to the 'Faulty Resource'.
12. 'Self-Configuration Manager' will remove the resource from the 'Active Resource' and will add it to the 'Faulty Resource'.
13. If 'Self Protection Manager' observes any kind of malfunction, it will call 'Self -Monitoring Manager' and will pass the resource id.
14. The 'Self-Monitoring Manager' will add the executing job in the resource reported by 'Self-Protection Manager 'to the 'Temp Job Queue' using the latest job checkpoint created by 'Monitoring Manager'.
15. 'Monitoring Manager' will then call the 'Self-Configuration Manager' to remove the resource reported by 'Self-Optimization Manager' from the 'Active Resource.
16. 'Self-Configuration Manager' will remove the resource reported by the 'Self-Protection Manager' and will add it to the 'Faulty Resource'.
17. If 'Monitoring Manager' receives an acknowledgement from the resource about the successful completion of the
18. job then it will pass the result of the executed to the 'Job Return Manager'.
19. The 'Job Return Manager' will return the job to the respective client.
20. 'Monitoring Manager' will now call the 'Self-Configuration Manager' to remove the resource from 'Active Resource'.
21. 'Self-Configuration Manager' will remove the resource from the 'Active Resource' and will add it to the 'Passive Resource'.
22. 'Self-Healing Manager' will be always checking for any resource in the 'Faulty Resource'. If it finds any, then it will remove all the threats from the resource.
23. 'Self-Healing Manager' will be always checking for any resource in the 'Faulty Resource'. If it finds any, then it will remove all the threats from the resource.
24. The 'Self-Protection Manager' will check that resource in the 'Faulty Resource' if it is free of any threats or not.
25. 'Self-Protection Manager' will return true if the resource is healed and false if it is not healed. If it returns false, the 'Self-Healing Manager' will again start healing the resource.
26. If 'Self-Protection Manager' returns true then 'Self-Healing Manager' will call the 'Self-Configuration Manager' to remove the resource from the 'Faulty Resource'.

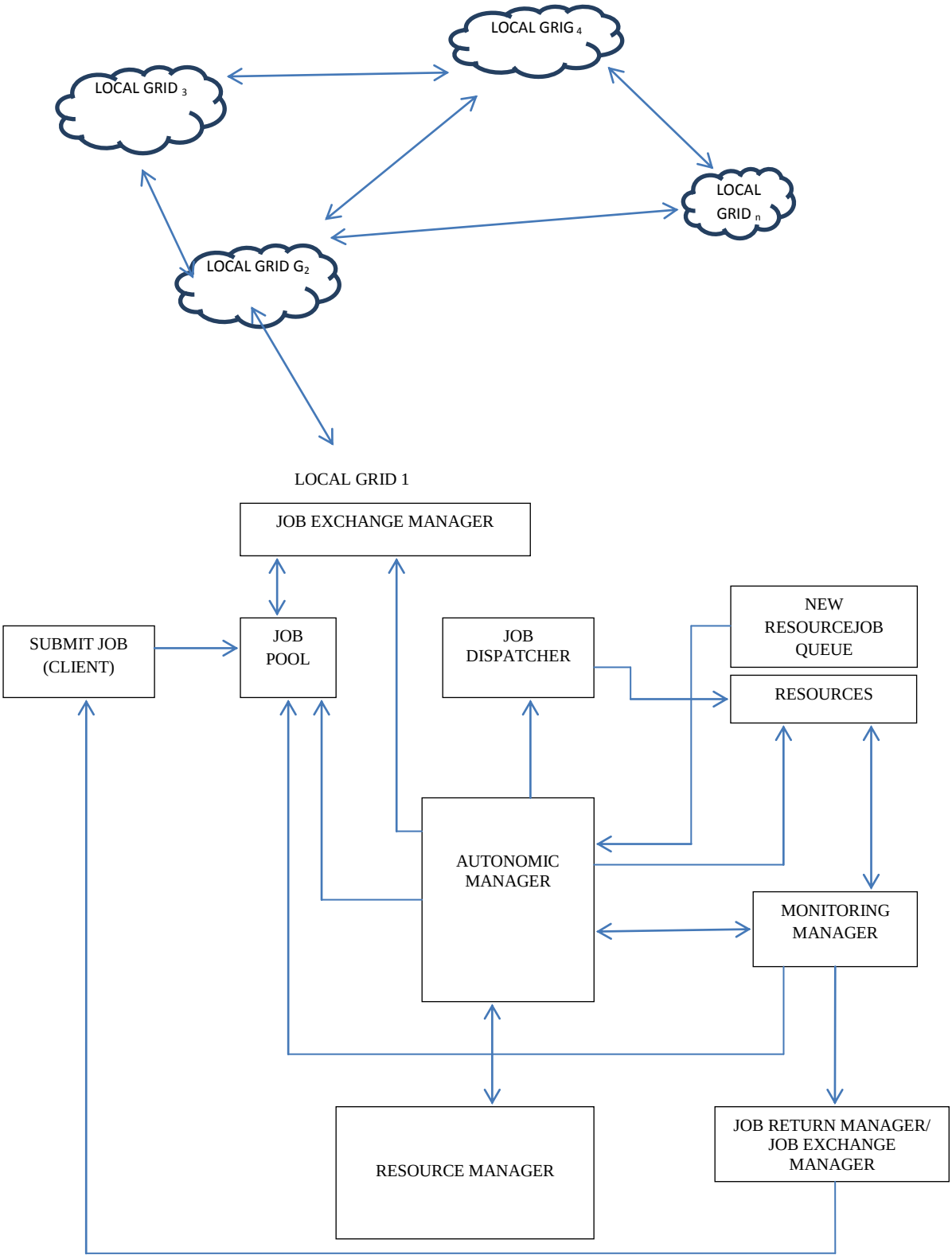

Figure 1: Grid connected network

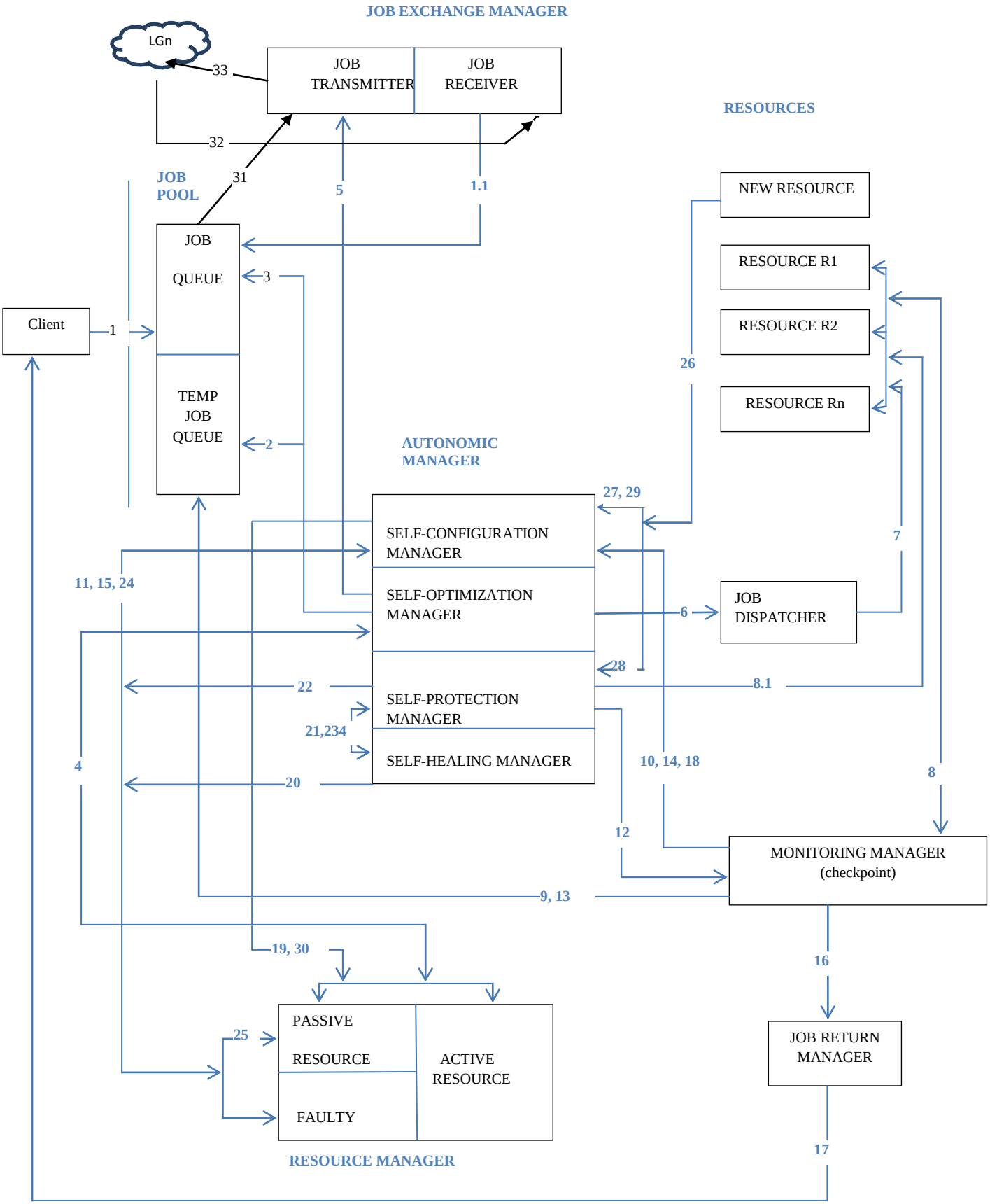

Figure 2: Proposed architecture

27. The 'Self-Configuration Manager will remove the resource from the 'Faulty Resource' and add it to the 'Passive Resource'.
28. If there is any resource who wants to join the grid then it will send a request to the 'Self-Configuration Manager'.
29. The 'Self-Configuration Manager' will call the 'Self-Protection Manager' to check the resource.
30. The 'Self-Protection Manager' will return either true or false to the 'Self-Configuration Manager'.
31. If 'Self-Protection Manager' will return false then 'Self -Configuration Manager' will discard the resource request to add it in the grid.
32. If 'Self-Protection Manager' will return true then 'Self-Configuration Manager' will add the resource to the 'Passive Resource'.
33. The 'Job Transmitter' will check the timestamp against each job in the 'Job Queue' and will send the job with the largest time stamp to the nearest local grid in the network.
34. The 'Job Receiver' in 'Job Exchange Manager' will receive the job sent by the nearest grid.
35. The 'Job Transmitter' will receive the job given by the 'Self-Optimizer Manager' in step 5 and will send it to the nearest local grid.

4.2 Algorithm for Protection Manager

1. Get all the resources in ACTIVE RESOURCE
2. Use Negative Selection Algorithm to Check Resources
3. If no threats detected then
   Return True
4. Else
   Return False
5. If return is equal to false
   Pass the resource id to Monitoring Manager

4.3 Algorithm for Self Healing Manager

1. Check the FAULTY RESOURCE for the resources
2. If Healed or Repaired
   Call SELF-PROTECTION Manager
   Call SELF- CONFIGURATION MANAGER to add the resource to PASSIVE RESOURCE
3. Else
   Call SELF CONFIGURATION MANAGER to remove the resource from RESOURCE MANAGER

4.4 Algorithm for Self Optimization Manager

1. If TEMP JOB QUEUE ! = NULL, then
   Check for resource in PASSIVE RESOURCE
   IF PASSIVE RESOURCE is equal to NULL
   CALL JOB TRANSMITTER
   Else
   Schedule the job using GENETIC ALGORITHM
   Call Job Dispatcher
2. Else
   If JOB QUEUE != NULL, then
   Check for resource in PASSIVE RESOURCE
   If PASSIVE RESOURCE IS equal to NULL
   CALL JOB TRANSMITTER
   Else
   Schedule the job USING GENETIC ALGORITHM
   Call Job Dispatcher
   Else
   Repeat step 1 to 2

4.5 Algorithm for Self Optimization Manager

This manager will add and remove a resource R from the Grid.

1. If a Resource send a request to autonomic manager
2. Call self-Configuration manager
3. Self-configuration manager then invokes Self-Protection manager which will check the resource R for any defects or threat. (so that it does not make the system unstable)
4. If resource R is good, then
   Return True to Self-Configuration Manager
5. Add the resource to the Passive Resources
6. Else
7. Discard the Resource R Request

4.6 Remove a resource from the Grid

1. If a resource sends a request to autonomic manager
2. Call configuration manager
3. If Resource R is in Passive Resource or Faulty Resource, then
   Remove the Resource R From resource manager
4. else
5. Configuration Manager will send the resource ID R to Monitoring Manager
6. Monitoring manager will add the partially executed job to the Temp job queue and Return TRUE
7. If TRUE
8. Self-configuration Manager will remove the Resource R from Resource Manager

4.7 If a resource R quits while a job execution is in progress

1. Monitoring manager will add the last checkpoint of the job executing in resource R to TEMP JOB queue.

Call self-configuration Manager to remove the resource from ACTIVE RESOURCE

5. CONCLUSION

Due to the inherent complexity, heterogeneous and dynamism of Grid systems, achieving large-scale distributed computing in Grids turns out to be an elaborated work. One way for this problem is using autonomic computing, which can change its behavior in response to changes in the status of the system. We have presented an architecture which merges Grid scheduling with automatic computing techniques in order to provide self-managing behavior.

In this project autonomic computing features is used in Grid to manage the grid more efficiently. We propose an autonomic Grid architecture as a possible solution, which can make decisions based on the current status of the system. Finally, all resources are used appropriately which in turn increases the performance of the Grid and at the same time the Grid security has been taken into consideration in this architecture. This architecture minimizes the risk of resource failure as the resources are being monitored continuously by the autonomic manager.

6. ACKNOWLEDGMENTS

Our special thanks to the Head of the Department of Computer science and engineering and Dean R&D Dr (Prof.) M.K.Ghosh of Sikkim Manipal Institute of Technology Majhitar for granting permission to use the infrastructural facility and Dr. (Prof.) C.T.Singh of Sikkim Manipal Institute of Technology, Majhitar for his guidance.

7. REFERENCES

[1] Chinmoy Kar, V.K.Rakesh, Tapas Samanta and S. Banerjee, 2012. A New Approach to Grid Scheduling using Random Weighted Genetic Algorithm with Fault Tolerance Strategy. International Journal of Computer Applications (0975–8887), Volume 48- No.23.

[2] Ebrahim Aghaei, Mohammad Saniee Abadeh, Mohammad Hossein Yektaie, , 2012. An adaptive sheduling system for computational grid using autonomic computing. International Journal of Computer Application(0975-888),volume 47-No.13.

[3] Nandagopal, M. & Dr. Uthariaraj, V.R. 2010. Fault tolerant scheduling strategy for computational grid environment.International Journal of Engineering Science and Technology. Vol. 2(9), 4361-4372.

[4] J. M. Schopf, 2003. Ten actions when grid scheduling ,in Grid resource management Management. State of theArt and Future Trends, first ed., Springer, pp. 15-23.

[5] M. Parashar, H. Liu and et al., 2006. AutoMate:Enabling Autonomic Applications on the Grid. ClusterComputing, vol. 9, no. 2, pp. 161-174.

[6] IBM White Paper, 2005. An Architectural Blueprint for Autonomic Computing. 3th ed., IBM Corporation.

[7] Garg, R, Singh, A.K. May, 2011. Multi-objective optimization to workflow grid scheduling using reference point based evolutionary algorithm, International Journal of Computer Application (0975-8887), Vol. 2(6).