A Fault Tolerance Service for QoS in Grid Computing*

Hwa Min Lee, Kwang Sik Chung[2], Sung Ho Jin[1], Dae-Won Lee[1], Won Gyu Lee[1], Soon Young Jung[1], and Heon Chang Yu[1]

[1] Dept. of Computer Science Education, Korea University, Seoul, Korea
{zelkova,wingtop,ldw1996,jsy,yuhc}@comedu.korea.ac.kr
http://comedu.korea.ac.kr

Abstract. This paper proposes fault tolerance service to satisfy QoS requirement in grid computing. The probability of failure in the grid computing is higher than in a tradition parallel computing. Since the failure of resources affects job execution fatally, fault tolerance service is essential in grid computing. And grid services are often expected to meet some minimum levels of quality of service (QoS) for desirable operation. However Globus toolkit does not provide fault tolerance service that supports fault detection service and management service and satisfies QoS requirement. In order to provide fault tolerance service and satisfy QoS requirements, we expand the definition of failure, such as process failure, processor failure, and network failure. And we propose fault detection service and fault management service and show simulation results.

1 Introduction

Grid computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and high-performance orientation [1]. Computational grid [2] consists of large sets of diverse, geographically distributed resources that are grouped into virtual computers for executing specific applications. As the number of grid system components increases, the probability of failure in the grid computing is higher than in a traditional parallel computing [2].

The vast computing potentiality of computational grids is often hampered by their susceptibility to failures, which include process failures, machine crashes and network failures. In grid computing, the fault management is very important and difficult problem to grid application developers. Since the failure of resources affects job execution fatally, fault tolerance functionality is essential in grid computing. Grid services are often expected to meet some minimum levels of quality of service (QoS) for desirable operation. Appropriate mechanisms are needed for monitoring and regulation system resource usage to meet QoS requirements [5]. Therefore, grid applications require fault tolerance services that detect resource failures and resolve from detected failures.

* This work was granted by University Research Program supported by Ministry of Information & Communication in republic of Korea.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2659, pp. 286–296, 2003.

However Globus toolkit [10] that is an integrated set of basic grid services does not provide fault tolerance service to satisfy QoS requirements. Although the Heartbeat Monitor [4] was an attempt to provide some support for fault detection, it has been discontinued as a separate module. HBM (Heartbeat Monitor) only detects process failure and computer failure through periodic heartbeat messages to a centralized collector. In this paper, we provide fault tolerance service that detects resource failures, deviations from required QoS levels, and excessive resource usages and resolves detected failures. In order to provide fault tolerance service and satisfy QoS requirements, we expand the definition of failures, such as process failure, processor failure, and network failure. And we propose fault detection service using fault detector, and fault management service using fault manager.

This paper is organized as follows: In Section 2, we present related work about fault tolerance service. Section 3 presents a system model, failure definition and architecture of fault tolerance service. Section 4 discusses fault detection service and fault management service. Section 5 shows simulation results. Finally, the paper concludes in Section 6.

2 Related Work

In Globus, a noticeable flaw is the lack of support for fault tolerance [7]. To date, grid applications have either ignored failure issues or have implemented fault detection and response behavior completely within the application [6]. The support for fault tolerance has consisted mainly of fault detection services and monitoring system. The HBM [4] designs and implements local monitor and data collector for providing fault detection service of process and computer for applications developed with the Globus toolkit. The local monitor locates on each host, and monitors processes' state on each host through periodic "I-am-alive" message. It arranges result of monitoring and generates heartbeats message periodically. And the data collector identifies failed computers based on missing heartbeat messages. However, the HBM can detect the limited failures, i.e., a process failure and a computer failure and did not provide fault manage service. While the NWS (Network Weather Service) [9] monitors available network bandwidth, memory, fraction of CPU available, free memory size, and free disk space size, the NWS can not provide fault detection service and fault management service. The Big Brother [12] is the monitoring, fault notification, and web-based system and network administration tools for distributed systems. But the Big Brother can not support application-specific event notification.

To support a fault tolerance service and QoS, fault detection service and fault manage service considering QoS requirements are essential. However, the Globus doesn't provide any one at present. Thus we propose fault tolerance service that detects failures through monitoring process, processor, and network and manages the detected failures. Our fault tolerance service also satisfies the QoS requirements of grid application.

3 A Design of a Fault Tolerance Service

We discuss our system model, the definition of failures, and the architecture of fault tolerance service.

3.1 System Model

In this paper, we propose the fault tolerance service for applications developed with the Globus toolkit. The proposed fault tolerance service supports Linux, Unix, and Solaris operating systems. In common grid computing, resource components are process, processors within a computer, network interfaces, network connections, specific computers, entire site, and specific computers. If we would consider all these resource components, the complexity and overhead of fault tolerance service increase. For reasons of complexity and overhead, we define the resource failure components as process, processor, and network. When a process may fail, it loses its volatile state and stops execution according to the fail-stop model. To provide fault tolerance service, we propose a fault detector and a fault manager. The fault detector is responsible for monitoring resources state and detecting resource failure. The fault manager is responsible for managing detecting failures. Fig. 1 shows modified GRAM [8] architecture when fault detector and fault manager is employed in the existing GRAM.

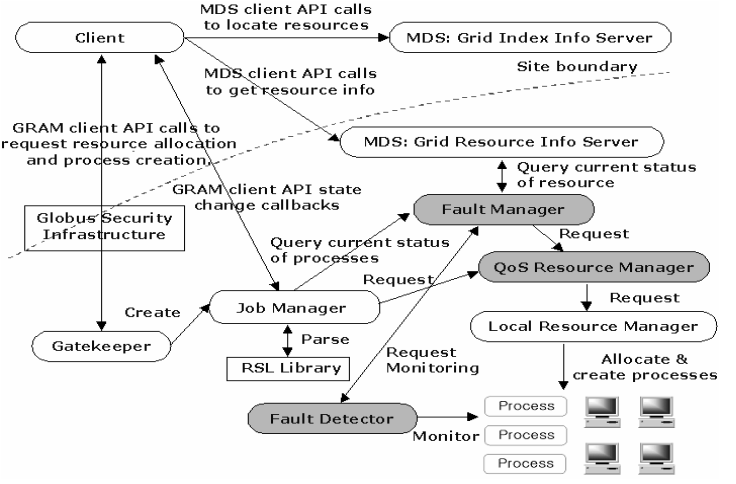


Fig. 1. A modified GRAM architecture when fault detector and fault manager is employed in the existing GRAM

3.2 Expansion of Failure Definition

In distributed computing, the resource sharing focuses on the process sharing. Thus the main target of fault tolerance is process in distributed computing and many fault-tolerance techniques for process have been studied in approach of checkpoint, message logging, and replication.

But computational grids shares processor, memory, storage, network and software as well as process. In computational grids, it is significant that fault tolerance service deals with various types of resources failure that dose not satisfy QoS requirements. Thus we expand definition of failure in order to provide fault tolerance service to satisfy QoS requirements.

[*Definition* 1] failure
It is a failure if and only if one of the following two conditions is satisfied.
1. resource stop due to resource crash
2. availability of resource does not meet the minimum levels of QoS

Table 1 shows the expanded failures that include process failure, processor failure, and network failure according to Definition 1.

Table. 1 Types of expanded failures

Process failure
1. a process stop (Process stop failure) 2. a starvation of process (Process QoS failure)
Processor failure
1. a processor crash (Processor stop failure) 2. a decrease of processor throughput due to burst job (Processor QoS failure)
Network failure
1. a network disconnection and partition (Network disconnection failure) 2. a decrease of network bandwidth due to communication traffic (Network QoS failure)

3.2 Architecture of Fault Tolerance Service

Figure 2 shows the architecture of proposed fault tolerance service in this paper.

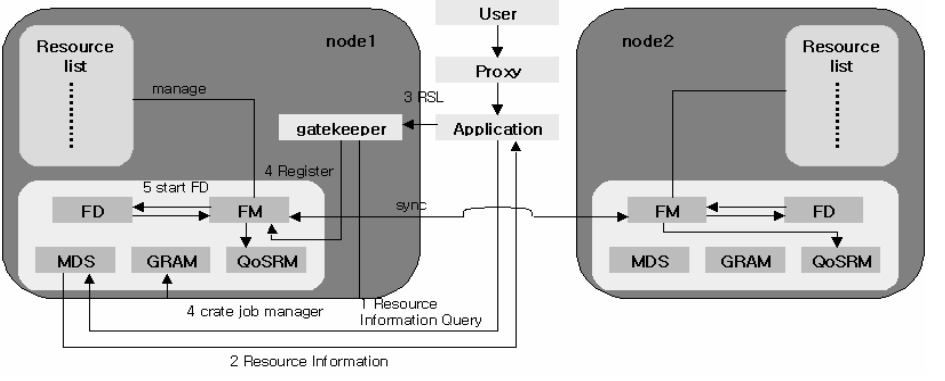


Fig. 2. Architecture of fault tolerance service

i) A fault detector (FD) is responsible for monitoring the states of process, processor, and network and deciding the occurrence of failures. The fault detector decides the occurrence of each resource failure from the detected state information and transmits the occurrence of resource failure to fault manager.

ii) A fault manager (FM) is responsible for displaying the states of resources and managing failures. The fault manager receives the state information of resources from fault detector and then displays the states of whole resources. And fault manager resolves the failures according to the recovery polices of each resource if failures occur.

iii) A QoS resource manager (QoS RM) provides advanced reservation and end-to-end management for QoS on various types of resources, such as processor, network, disks, and memory.

iv) GRAM (grid resource allocation management) [8] is responsible for allocation of computation resources and control of computation on those resources.

v) MDS (meta computing directory service) [9] provides a configurable information provider component called a Grid Resource Information Service (GRIS) and a configurable aggregate directory component called a Grid Index Information Service (GIIS).

vi) A resource list keeps the information of resources that the running jobs use.

In figure 2, the procedure of fault tolerance service is as follows. An application requests resource information to MDS and then MDS returns requested resource information. The application makes ground RSL using received resource information and requests resource allocation and process creation through gatekeeper. The gatekeeper authenticates mutually user and resource and starts a job manager that executes as that local user and actually handles the request. Simultaneously the gatekeeper registers resources information to fault manager. A fault manager requests monitoring of resource state to fault detector and then fault detector periodically reports resource state information to fault manager. The fault detector decides occurrence of failure through analyzing resource state information and transfers failure information to fault manager. If fault manager receives failure information, fault manager resolves failure according to the management policy of each resource.

4 Fault Tolerance Service

Figure 3 shows the components of fault detector and fault manager.

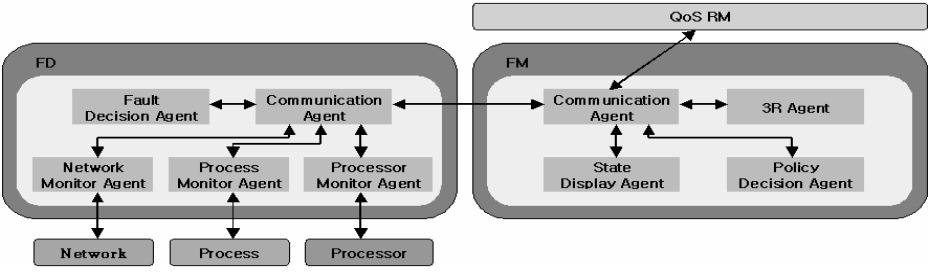


Fig. 3. Components of fault detector and fault manager

4.1 Fault Detection Service

Fault detector provides fault detection service that is as follows:

- Monitoring service, which monitors resource states of process, processor, and network
- Fault detection service, which decides the failure occurrence for each resource
- Communication service, which provides communication with each component

For fault detection service, the fault detector consists of process monitor agent, processor monitor agent, network monitor agent, fault decision agent, and communication agent.

i) A process monitor agent monitors process' state and starvation of process in job scheduler queue. The process monitor agent divides process' state into processing state, stop state, silent state, and unknown state.

ii) A processor monitor agent monitors process crash state and normal state. During processor's normal execution, processor monitor agent collects the used CPU utilization and the available CPU utilization.

iii) A network monitor agent monitors communication bandwidth, communication latency time, and network disconnection and partition between its own node and connected nodes.

iv) A fault decision agent decides the occurrence of failure through analyzing states information of each resource and then it identifies process failure, processor failure, or network failure. If failure occurs, fault decision agent reports failure information to fault manager. Fig.4 shows the failure decision algorithm of fault decision agent.

```
Fault Decision Agent
if receive resource_state_information message from communication agent
  then call DCT_OccurFailure();
Failure_state = OccurFailure; send Failure_state message to communication agent;

DCT_OccurFailure (CPU_state, Network_state, Process_state,
  Available_CPU_Utilization, Network_Bandwidth, Process_Waiting_Time)
{ if (CPU_state==failed) then return OccurFailure=yes;
  if (Network_state=disconnected) then return OccurFailure=yes;
  if (Process_state==suspend or Process_state=unknown)
    then return OccurFailure=yes;
  if (Available_CPU_Utilization < required_CPU_Utilization or 0 <= Available_
    CPU_Utilization <=20) then return OccurFailure=yes;
  if (Network_Bandwidth < requied_Network_Bandwidth)
    then return OccurFailure=yes;
  if (Process_Waiting_Time > Therhold_Waiting_Time)
    then return OccurFailure=yes;
  else return OccurFailure=no; }

Communication Agent
if receive Failure_state message from fault decision agent then if (Failure_ state=yes)
  then send resource_state_information message to policy decision agent;
```

Fig. 4. Fault decision algorithm

If states information of process is S (silent) or U (unknown), the fault decision agent detects the process stop failure. When fault decision agent receives the processor crash information, it detects the processor stop failure. If the CPU utilization is less than required quality of application or the CPU utilization is 0 ~ 20%, fault decision agent detects the processor QoS failure. And when fault decision agent receives the network disconnection or partition information, it detects the network disconnection failure. And if the amount of available network bandwidth is less than required quality of application or there is too much network traffic, fault decision agent detects the network QoS failure.

v) A communication agent manages communication between agents, FM and FD.

4.2 Fault Management Service

Fault manager provides the fault management service as like follows:

- Display service, which displays each resource state to user
- Policy service, which applies fault management method based on predefined policy according to kinds of resource failure
- Communication service, which provides communication with each component

```
Policy Decision Agent
if receive failure_state_information from communication agent then call CBR();
optimal_resource_allocation = best solution;
send optimal_resource_allocation message to communication agent;

CBR(failure_state_information)
{   retrieve best_matched_cases from Case_Library;
    while(evaluation_value >= threshold){ adapt best solution to current situation;
      evaluate evaluation_value; modify best solution; }
   apply best solution in real world; evaluate apply_results;
    add current case and best solution to Case_Library;  }

Communication Agent
if receive optimal_resource_allocation from policy decision agent
  then  send resource_reallocation_request message to 3R agent;
```

Fig. 5. Policy decision algorithm

For fault management service, the fault manager consists of state display agent, policy decision agent, resource reallocation request agent, and communication agent.

i) A state display agent displays each resource state to user using resource state information from fault detector and identified failure information from fault decision agent.

ii) A Policy decision agent examines predefined condition for failure management, when fault decision agent identifies QoS failure of each resource. Then, policy decision agent decides whether it requests job reallocation to QoS RM or continues doing

the job at current node. Figure 5 shows the policy decision algorithm of policy decision agent.

iii) A 3R(resource reallocation request) agent requests job reallocation of QoS RM for failure recovery if process stop failure, processor crash, and network partition occur, and policy decision agent decides job reallocation for QoS failure recovery.

iv) A communication agent manages communication between agents, FM and FD, FM and QoS RM.

5 Simulation

In this paper, the goals of simulation experiment are as follows:

i) The fault detector monitors the resource state of process, processor, and network.

ii) The fault detector decides failure occurrence and identifies type of failure by monitoring resource state information.

iii) When failure occurs, fault manager decides which policy for failure recovery.

Figure 6 shows MDS structure for fault tolerance service simulation. Our testbed consists of the six hosts for simulation.

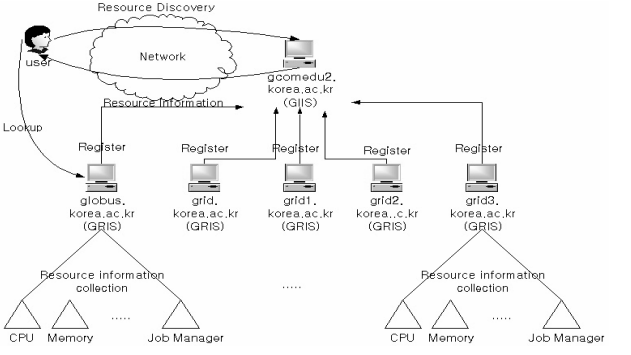


Fig. 6. MDS structure for simulation

For simulation of the fault tolerance service, we developed the simulation system with JAVA. The simulation system shows process state, processor state, and network state per a second. The features of our simulation system are as follows:

First, our simulation system displays the real-time information of the resource state information. The displayed information is running process' state, CPU utilization, network bandwidth between nodes. Second, our simulation system decides the QoS failure occurrence if resource state does not satisfy QoS requirement. And it notifies the failure occurrence to user.

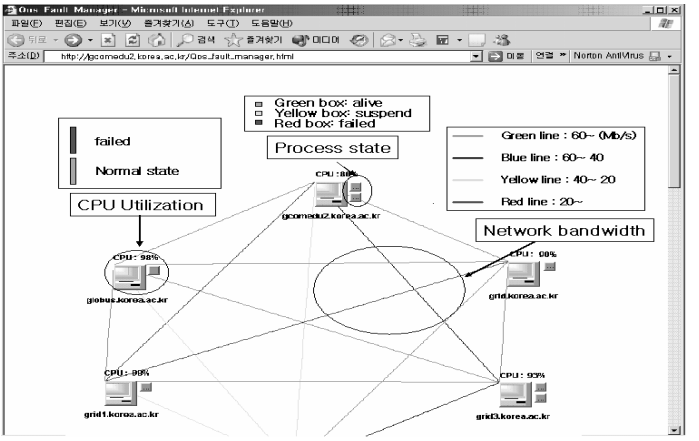
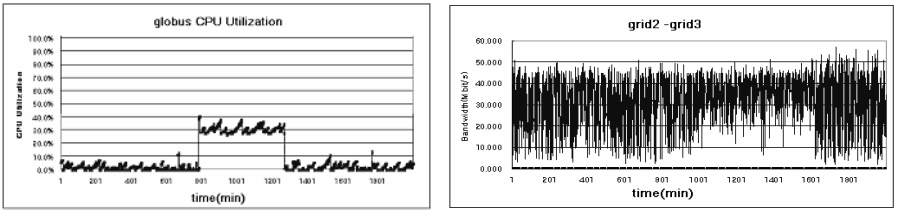


Fig. 7. The interface of our simulation system

Figure 7 shows the interface of our simulation system. Our simulation system supports GUI in order to check resource state easily. In Fig. 7, computer icon symbolizes the each component of MDS. CPU utilization is displayed by percentage (%) above computer icon. The processor state is represented with vertical bar at the right side of computer icon. If processor failure occurs, the color of vertical bar turns red. If the color of vertical bar is green, it means normal state. The network bandwidth between nodes is shown by colored line. If the color of line is green, it means 60Mbps or more. If the color of line is blue, it means 20 ~ 40Mbps. If the color of line is red, it means 20Mbps or less. The process state is represented with small box at the right side of processor state bar. If the color of small box is green, it means normal state. If the color of small box is yellow, it means suspended state. If the color of small box is red, it means failed state.



(a) CPU utilization (b) Network bandwidth

Fig. 8. CPU utilization state information and network bandwidth state information

In simulation system, we monitored resource states in components of our testbed for 2000 minutes. Figure 8a shows CPU utilization state information, one of the six hosts in our testbed. In Fig. 8a, the CPU utilization is increased rapidly from 800 min to 1270 min. At that time, we detect CPU QoS failure through simulation application and select predefined policy for resolving failures. Figure 8b shows network band-

width state information between two nodes. Dynamism of network bandwidth is relatively reduced from 1000 min to 1600 in Fig. 8b.

We receive notification of network QoS failure through application, and execute predefined policy for resolving failures. But in case of process failure, we do not detect a native fault during monitoring. So we involve the failure injection through process kill on Unix System signal. Through simulation results, we verified that the proposed fault tolerance service monitors the resource state information of process, processor, and network, decides failure occurrence and identifies type of failure by monitoring resource state information. We also verified the need and usefulness of QoS fault tolerance service.

6 Conclusion

The vast computing potentiality of computational grids is often hampered by their susceptibility to failures, which include process failures, machine crashes and network failures. And grid services are often expected to meet some minimum levels of quality of service (QoS) for desirable operation. Therefore, grid applications require fault tolerance services that detect resource failures and resolve detected failures. Appropriate mechanisms are needed for monitoring and regulation system resource usage to meet QoS requirements. In this paper, we proposed fault tolerance service for process, processor, and network. In order to provide fault tolerance service and satisfy QoS requirements, we expanded the definition of failures and proposed fault detection service and fault management service. And we verified our fault tolerance service through the simulation.

In future work, we will implement our fault-tolerant service to satisfy QoS requirement in Globus toolkit. And we will implement the QoS resource manger for fault management service and quality of service on various types of resources.

If you wish to include color illustrations in the electronic version in place of or in addition to any black and white illustrations in the printed version, please provide the volume editors with the appropriate files.

References

1. I. Foster, C. Kesselman, S. Tuecke, The Anatomy of the Grid : Enabling Scalable Virtual Organizations, International J. Supercomputer Applications, 15(3), 2001.
2. Ian Foster, Carl Kesselman, The Grid : Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, 1998.
3. I. Foster, A. Roy, V. Sander, A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation, 8th International Workshop on Quality of Service, 2000.
4. P. Stelling, I. Foster, C. Kesselman, C.Lee, G. von Laszewski, A Fault Detection Service for Wide Area Distributed Computations, Proc. 7th IEEE Symp. on High Performance Distributed Computing, pp. 268–278, 1998.

5. A. Waheed, W. Smith, J. George, J. Yan, An Infrastructure for Monitoring and Management in Computational Grids, In Proceedings of the 5th Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers, March, 2000.
6. Anh Nguyen-Tuong, Integrating Fault-Tolerance Techniques in Grid Applications, Ph.D. Dissertation, August, 2000.
7. R.J. Allan, D.R.S. Boyd, T. Folkes, C. Greenough, D. Hanlon, R.P. Middleton, R.A. Sansum, Globus and Associated Grid Middleware, Consolidated Evaluation Report from UKHEC Sites, 2001.
8. K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, A Resource Management Architecture for Metacomputing Systems, Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, pg. 62–82, 1998.
9. M. Swany and R. Wolski, Representing Dynamic Performance Information in Grid Environments with the Network Weather Service, 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), Berlin, Germany, May 2002.
10. I. Foster, C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit, Intl J. Supercomputer Applications, 11(2): 115–128, 1997.
11. I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy, A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation, International Workshop on Quality of Service, 1999.
12. Big Brother System and Network Monitor, available from `http://maclawran.ca/bb-dnld/bb-dnld.html`