*A Scalable Service Discovery and Resource Assignment Scheme for Network Computing Systems

ManChon U[1,2] **Shen Wenfeng[2]** **Xu Weimin[2]** **Zhu Yonghua[2]**

[1]Av. Horta E Costa 15, 5-Andar-D, Ed. LeiKuan, Macau

[2]Department of Computer Science and Engineering, Shanghai University, 200072, Shanghai, China

Email: manchonu@yahoo.com , Tel: +853-66677159

Abstract – *With the ongoing development of network computing, it is crucial to able to schedule and manage the available computing resources accurately and efficiently. In this paper, we present a scalable service discovery scheme and resource assignment strategy for network computing systems. We first introduce the architecture of task scheduling and resource management system followed by a discussion of the approaches needed to ensure the validity of resources and fault tolerant computing. Furthermore, the experiment is done to demonstrate the scalability, robustness and efficiency of service discovery scheme and resource assignment strategy as proposed. Finally, throughout the course of the experiment, we compare the performances of different resource scheduling algorithms based on different assigning policies.*

Keywords: Network computing, Service discovery, Pervasive computing, Fault Tolerant computing, Task Scheduling Algorithm

1. Introduction

Recently, scientists began investigating on how to improve the performance of various communication protocols armed with the knowledge that computer networks often have a scale-free structure. For example, researchers are improving the Gnutella peer-to-peer protocol [1] which has previously been shown to scale very poorly [2] by altering the protocol to exploit the scale-free topology of the peers. Anthill is a framework for peer-to-peer computing that makes use of network computing systems theory to provide scalability and adaptability.

This paper shows the scalable service discovery scheme and resource assignment strategy for Pervasive computing environments. Network Computing belongs to the Pervasive computing environment, which utilizes idle CPU cycles and storage space of hundreds or thousands of networked systems to work together on a particularly processing-intensive problem. It solves a large problem by giving small parts of the problem to many computers to solve and then combining the solutions for the parts into a solution for the problem [3].

A Cluster is a group of computers which act as a whole to provide computing services to users [4]. When we use cluster for parallel computing, we discover that many tasks with high serialization did not need to be processed by using the expensive special parallel computing devices. Hence we developed Service Oriented Infrastructure for Network Computing (SOINC), which is a system working between numerous computers and the operation is dependent on the network. It has to be able to conquer the heterogeneous environmental problem. Normally, environmental heterogeneity consists of two aspects: heterogeneity of operating system and different location.

*This project is supported by E-Institute of Shanghai Municipal Education Commission – Shanghai High Education Grid (200312) and by the Natural Science of Shanghai Education Commission (Project 03AK70)

The rest of this paper is organized as follows. In Section II, the SONIC architecture is introduced. In Section III, the task scheduling and resource management system is presented. In Section IV, the results of the experiments are discussed. Finally, conclusions are presented in Section V.

2. SOINC Architecture

The entire SOINC system consists of three parts (Figure 1). The Workstation is used by Subscribers for developing, submitting, controlling the tasks and managing the data. The Server takes responsibility for corresponding all of the operations of Network Computing Service. The Computing Node offers available computing resources.

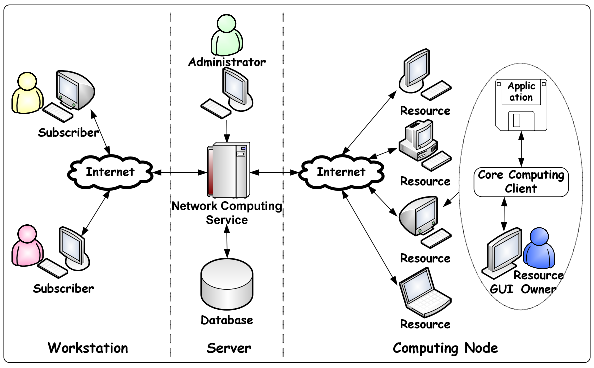


Figure 1. SOINC Architecture

Some basic concepts of SOINC are stated as follows：

(1) Service. Service is one of the crucial concepts of SOINC; development of the whole SOINC is surrounding the different operational services. Aggregate which provides computing ability on certain extent and related equipment would be encapsulated as a service. For instance, a computer with SOINC Client installed can be encapsulated as a service [5,6,7].

(2) Community. Community plays the role of a resource pool in SOINC. All functions in SOINC system can be encapsulated as service abstractly [5,6,7]. Resources which are encapsulated as service can register into the community at any time.

(3) Registry. In the distributed model of SOINC, firstly, the Service Provider (Computing Node) informs the SOINC Registry of its presence, its location and some of its information by registering with the SOINC Registry. Afterwards, the SOINC Registry puts that computing node (R1) into the community as an available resource, and then gives R1 an Expiration Time (T1); R1 will be ranked to invalidate if it has not updated its expiration time before certain checkpoint T2 (T2>T1) of the community. Furthermore, the task which had been assigned to R1 will be reassigned [5,6,7] (Figure 2).

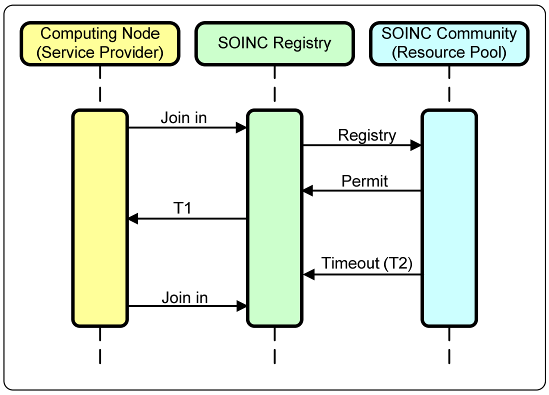


Figure 2. Registry Mechanism of SOINC

(4) Subscriber. The Subscriber is the editor of task and the consumer who utilizes this computing service ultimately. A suit of Network Computing Service should be able to provide services to many subscribers as though the usage manner is that of a perfect supercomputer.

(5) Client. Client refers to the computing node, which can be the personal computers provided by volunteers on the Internet, or the amateurish desktop computers provided by certain organizations as well as part of the appropriative nodes of a supercomputer. The SOINC Client software should be deployed on these computing nodes where the resource owner possesses the absolute right of controlling the software and the machine.

3. SOINC Scheduling & Resource Management System(SS&RMS)

SS&RMS mainly takes four roles in SOINC system, as states below:

(1) Manage the requests received and the

processing of computing tasks

(2) Manage the Join in, Drop out and Availability of computing resources

(3) Schedule the available computing resources

(4) Manage the Feedback and Fault Tolerance of computing resources

Figure 3 shows the simple operating architecture of SS&RMS, which describes the operating theory and architecture of SS&RMS, further illumination of every parts will be made as following:

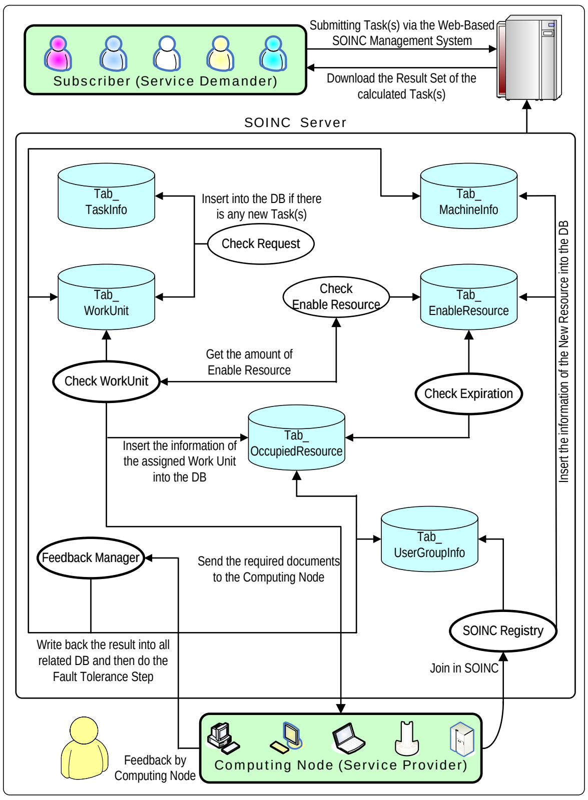


Figure 3. SS&RMS Architecture

3.1. Manage computing tasks

This part mainly takes charge of receiving tasks that are submitted by service demanders (Subscribers). After starting SS&RMS, a thread, which detects the arrival of tasks, will start and wait for activation by the Web-based Management System (WMS) of SOINC. The Subscriber sends the task and all relevant documents to SS&RMS via WMS; SS&RMS receives the task and documents and proceeds to write all relevant information into the database; after which the task waits to be assigned. Every task submitted by the service demander may contain several Work Units. In this article, we assume that every task received by SS&RMS is already divided into several work units.

3.2. Manage Dynamic Resources

This part chiefly takes charge for the Join in, Drop out and checking availability of computing resources provided by service provider.

In order to join in SOINC, the resource owner has to have a unique account ID of SOINC user group; if the resource owner does not have an account ID, he/she can register an account via the WMS. If it is the first time the computing resource is joining SOINC then SS&RMS will generate a unique machine ID to the computing resource. Finally, after the verification of login information, SS&RMS will send back the confirmable information, which contains the machine ID and expiration time, to the computing resource (Figure 4).

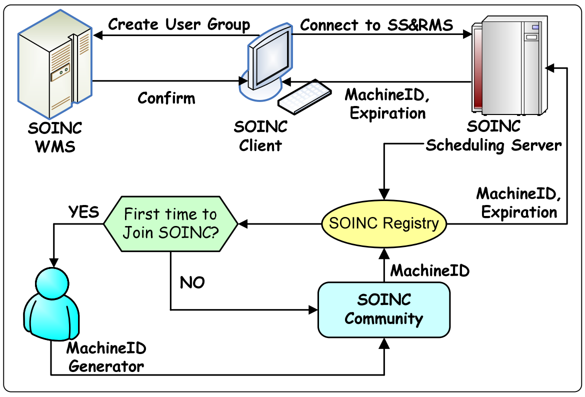


Figure 4. Computing resource join in SOINC

In order to make good use of resources and considering the load balance of the server, computing resources dropping out from SOINC Community will not be detected by SS&RMS immediately, but gives this responsibility to an inspecting thread, which is used for checking the availability of resources. The inspecting thread makes the corresponding transaction after checking the availability of all resources that exists in SOINC community between a stable time slice (Figure 5). By reason of computing resources might be dropped out of the SOINC community temporary because of such exceptional situations as network or hardware problems, but then return to join in the SOINC community to update its expiation time in a short time period after restarting the machine or reconnecting the network. Therefore there is no influence for the calculation of tasks which are assigned to these sorts of computing resources. Hence the immediate detection of resources dropping out is not necessary.

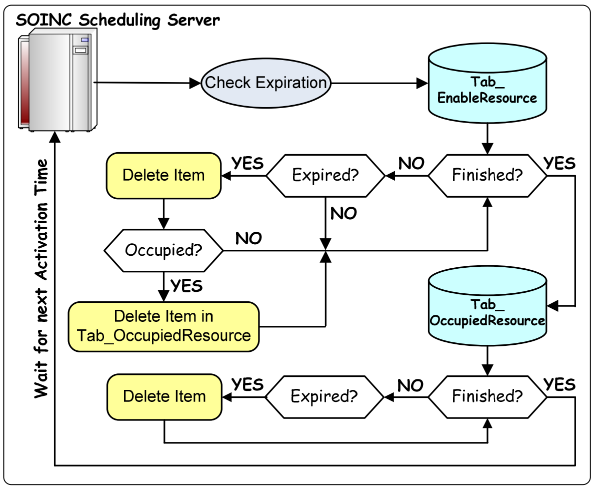


Figure 5. Check availability of resources

3.3. Scheduling Computing Resources

To cooperate with the Fault Tolerance Mechanism, every work unit will be assigned to three computing nodes for calculation in our scheduling algorithm, shown below.

Scheduling Algorithm of SS&RMS

```
start scheduling thread()
   i=0
   repeat
      try to do following:
         while (exists available resource)
            sort_workUnit(Tab_WorkUnit)
            sort_enableResource(Tab_EnableResource)
            loopCount=3-assignedNodes
            while (i<loopCount)
               archiveTransporter(source_file)
               if (transfer failed)
                  transferFlag=false
                  break
               end if
               assign(enableResource, workUnit)
               i++
            end while
         end while
         wait for a second
      exception:
         Printout the Error Message
   until the end
end scheduling thread()
```

According to the algorithm above, after the startup of the scheduling thread, the computing nodes will be assigned to corresponding tasks as long as there remains available resource in the SOINC Community. Otherwise it will break out of the loop and wait for a second to buffer, then, checks the existing status of resource afresh. If the condition is satisfied, the program sorts the list of work units which are waiting to be assigned by certain priorities, then sorts the list of available resources by score from high to low, and lastly begins the assigning. Every work unit is assigned to three computing nodes to cooperate with Fault Tolerance Mechanism. The scheduler will choose the available resource with the highest score in the list of available resources (the first item of the sorted list) for every assignment so as to improve the possibility of finishing the calculation of the tasks, since a higher score leads to a higher reliability of the computing resource. If the exceptional situation of invalid resource occurs during the scheduling, i.e. the computing resource which was in the SOINC Community might be dropped out before this scheduling moment (Shut down, Network problems, etc.), but the invalid resource has not been eliminated since the time period of certain checkpoint has not come yet before this scheduling moment. Then it breaks out of the loop after eliminating the invalid computing resource and then reassigns the task to other resources or waits for the advent of new resources.

3.4. Fault Tolerance

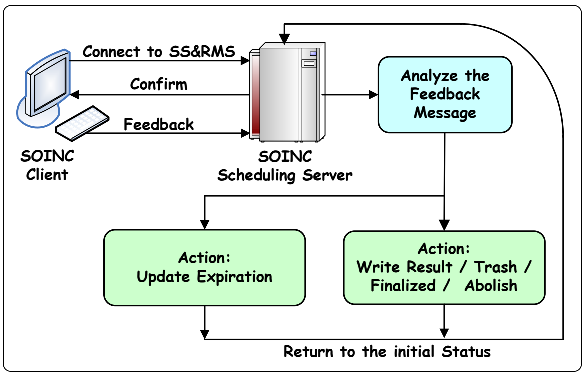


Figure 6. Manage feedback messages

This part primarily takes the responsibility of receiving the Feedback from service providers (Computing Node) and operating the process of Fault Tolerance. Events activated by the message which are feed backed by service providers could be separated into follows: “Update Expiration”, “Write Result”, “Trash”, “Finalized” and “Abolish”, Steps of operation of feedback mechanism refers to figure 6.

The first sort of feedback message activates the thread of updating expiration time, whereas the other four feedback messages activate the thread of fault tolerance mechanism. Their functional definitions are as following:

Write Result – To write the result which is feed backed by computing resource into corresponding table of Database. This event will only be activated in case the corresponding work unit is not finalized at the moment.

Finalized – To finalize the result of the relative work unit, then calculate scores for corresponding computing nodes. If there are two or more results which are feed backed for the same work unit and the two results are equal to each others, then we assert that the result is right.

Trash – If the corresponding work unit was finalized before this feedback message, then we trash the feed backed result without writing into the database after calculating the score for the computing node.

Abolish – If three of the feed backed results for the same work unit are different, then SS&RMS puts the work unit into the waiting list after calculating the scores for the computing nodes in order to wait for reassigning.

Steps of operation of fault tolerance mechanism refer to figure 7.

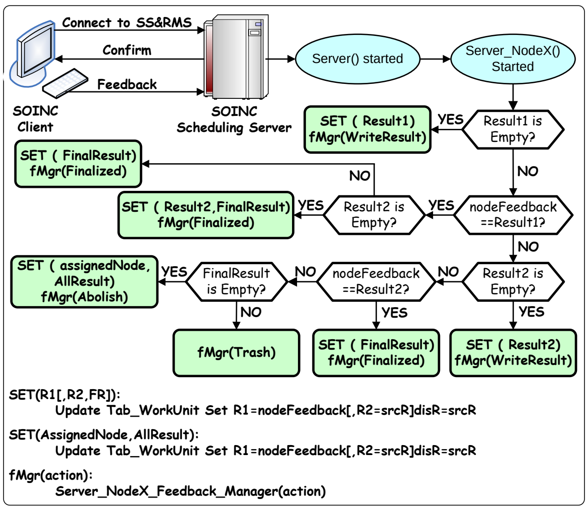


Figure 7. Fault Tolerance Mechanism

4. Simulation Results

We simulated 100 computing nodes to execute 1000 work units (every work unit assigns to three computing nodes) in our experiments. Each work unit is associated with different orders of priority, entry time and size of calculating documents in the same conditions. We randomly generated the priority between 1 and 5, and the size of calculating documents of corresponding work unit is generated by a random function as “random.nextInt(20067)+random.nextInt(1984)+random.nextInt(1023)” which generate an integer with the unit of kilobyte.

Figure 8 shows the result of performance comparison between different amounts (150 and 300) of available computing resources.

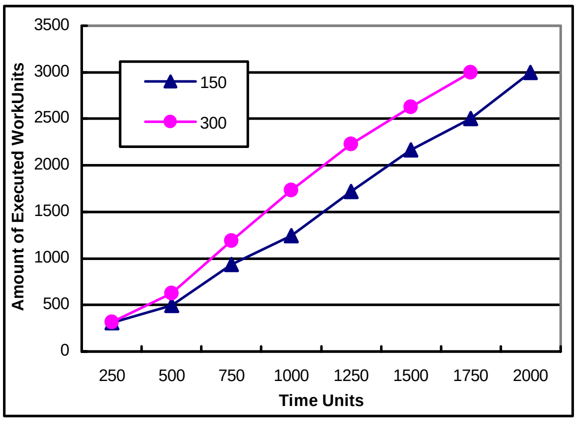


Figure 8. System Performance with different amount of computing nodes

Figure 9 shows the difference in performance between Pipelining assignment and Unification assignment in this situation, there are limited computing resources but numerous tasks. Unification assignment means that if there are just two available resources in the SOINC Community. But the system regulates that every work unit has to be assigned to three computing nodes. Then the system stops assigning and waits for the advent of the third computing nodes, where after assigns the work unit once. Pipelining assignment is used by this system for assigning which will be carried out if only there is just one available computing node in the SOINC Community.

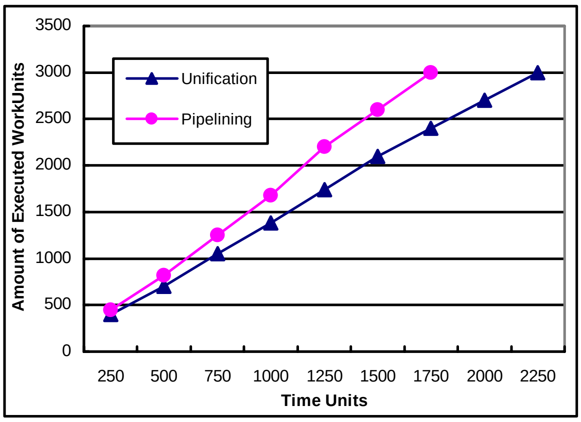
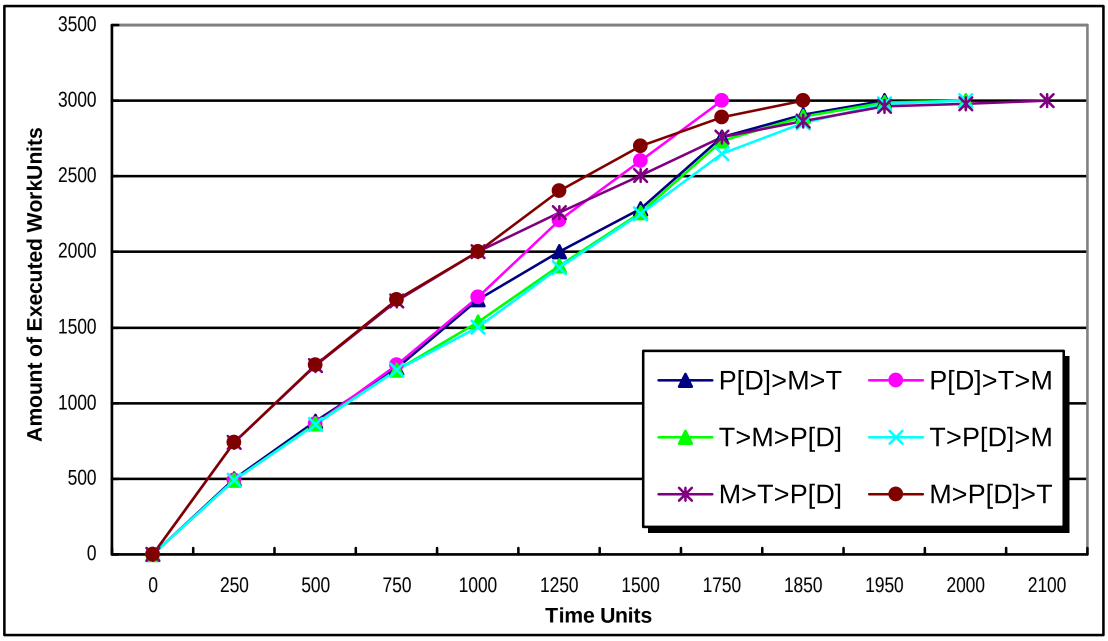


Figure 9. Pipelining assignment versus Unification assignment

Figure10 illustrates all the possible combinations between P[D], T and M, and the performance of each, where "P[D]" represents priority of the work unit, "T" represents the entry time of the work unit when it is submitted and "M" represents size of calculating documents of the work unit. As illustrated, the fastest and most stable performance of the six types would be "P[D]>T>M".Since we have imported the relative expiration time of tasks, which is constraining that task should be finished calculating before corresponding expiration time. If the computing node could not finish calculating the task assigned to it before the task's expiration time, then that task will be reassigned because it was invalidly calculated. Therefore, we can use the task's expiration time instead of the priority "P[D]" in respect that this approach not only calculates the priority but also debases the amount of reassigning tasks which was caused by missing the task's deadline. The study of deadline scheduling has been discussed in [8,9], and also has proved the reason of using deadline as highest priority for tasks' scheduling. We added the "T" and the "M" mentioned above. Then we got the best scheduling algorithm by ourselves finally, as if figure 10 shows.



	0	250	500	750	1000	1250	1500	1750	1850	1950	2000	2100
P[D]>M>T	0	501	879	1231	1683	2003	2285	2759	2908	3000		
P[D]>T>M	0	491	865	1253	1700	2210	2604	3000				
T>M>P[D]	0	492	862	1219	1535	1910	2262	2733	2893	2988	3000	
T>P[D]>M	0	490	861	1220	1504	1897	2250	2650	2851	2979	3000	
M>T>P[D]	0	739	1250	1677	2002	2261	2506	2758	2867	2963	2980	3000
M>P[D]>T	0	740	1254	1683	2001	2404	2701	2892	3000			

Figure 10. Simulation Results

5. Conclusion and Future Works

We have constructed a Service Oriented Infrastructure for Network Computing (SOINC) System. By observing the experiment implemented above, SOINC could finish scheduling tasks and managing resources successfully and effectively. Whenever there is a new computing node that wants to join in our system, SOINC can detect it at once and then put it into SOINC Community as available resource. When the invalidation of resource occurs, SOINC could remove the invalid resource and reassign the task to other resources or wait for the advent of new resources. Furthermore, we have implemented the scoring mechanism of computing nodes to cooperate with the scheduling in order to ensure the validity of computing resources. Finally, we have implemented Fault Tolerance mechanism on SOINC effectively, which assures the correctness of computing tasks.

In the coming stage, we will pay more attention to the security of our system to prevent attacks from the baleful users.

References

[1] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernado A. Huberman. Search in power-law networks.Physical Review E, 64, 2001.

[2] Jordon Ritter. Why Gnutella Can’t Scale. No, Really. Technical report, Darkridge Security Solutions, Available from: http://www.darkridge.com/jpr5/doc/gnutella.html, 2001.

[3] David P Anderson. BOINC: A System for Public-Resou0rce Computing and Storage [J]. GRID, 2004, 4~10

[4] Xu Wei-min，Yan Yun-Zhong. Computer System and Architecture (Second Edition) [M]. Publishing House of Electronics Industry，July-2003.

[5] ManChon U. Open Infrastructure for Network Computing – Scheduling and Resource Management System [D]. Department of Computer Science and Engineering in Shanghai University, 2006.

[6] Rong Ye, Wen-Jing Hsu, Zehua Liu and Voon-Yee Vee. Jini-Enabled High Performance Computing[J]. In Proceedings of the 15th International Conference on Information Networking (ICOIN 2001) , pages 145-152

[7] Zoltan Juhasz, Arpad Andics, Szabolcs Pota. JM: A Jini Framework for Global Computing[J]. 2nd International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems at IEEE International Symposium on Cluster Computing and the grid (CCgrid'2002) May 21 - 24, 2002 Berlin, Germany.

[8] A. Takefusa, H. Casanova, S. Matsuoka, and F. Berman. A Study of Deadline Scheduling for Client-Server Systems on the Computational Grid[J]. Proceedings of 10th IEEE International Symposium on High Performance Distributed Computing (HPDC- 10), 2001

[9] Eddy Caron, Pushpinder Kaur Chouhan, Frederic Desprez. Deadline scheduling with priority for client-server systems on the Grid[J]. Grid Computing 2004. IEEE International Conference On Grid Computing. Super Computing 2004, October 2004

Correspondence

ManChon U received a B.Eng. degree in Computer Science and Technology from Shanghai University in 2006. His Research interests include distributed systems, parallel processing, network security and high performance computing.

E-mail: manchonu@yahoo.com

Tel : +853 – 66677159

CV : http://home.macau.ctm.net/~kevinu/CV.htm