Efficient Server Virtualization using Grid Service Infrastructure

Sung-Jin Baek*, Sun-Mi Park*, Su-Hyun Yang*, Eun-Ha Song* and Young-Sik Jeong*

Abstract—The core services in cloud computing environment are SaaS (Software as a Service), Paas (Platform as a Service) and IaaS (Infrastructure as a Service). Among these three core services server virtualization belongs to IaaS and is a service technology to reduce the server maintenance expenses. Normally, the primary purpose of sever virtualization is building and maintaining a new well functioning server rather than using several existing servers, and in improving the various system performances. Often times this presents an issue in that there might be a need to increase expenses in order to build a new server.

This study intends to use grid service architecture for a form of server virtualization which utilizes the existing servers rather than introducing a new server. More specifically, the proposed system is to enhance system performance and to reduce the corresponding expenses, by adopting a scheduling algorithm among the distributed servers and the constituents for grid computing thereby supporting the server virtualization service. Furthermore, the proposed server virtualization system will minimize power management by adopting the sleep severs, the subsidized servers and the grid infrastructure. The power maintenance expenses for the sleep servers will be lowered by utilizing the ACPI (Advanced Configuration & Power Interface) standards with the purpose of overcoming the limits of server performance.

Keywords—Server Virtualization, Grid Service, Grid Infrastructure, Power Efficiency, Cloud Computing

1. INTRODUCTION

Highly developed systems are required for the massive information processing which occurs in our knowledge and information society, and a number of servers are necessary to serve this purpose. These additional servers have led to a rapid increase in the maintenance expenses of IT infrastructures. The server increase has generated an additional layer of expense regarding extra server installations, acquisition of server installation space, labour expense increases for sever maintenance and extra expenses for sever maintenance power. The core service technologies involved in reducing the maintenance expenses for this cloud computing IT infrastructure are PaaS (Platform as a Service), SaaS (Software as a Service) and IaaS (Infrastructure as a Service) [1]. Server virtualization is a service technique belonging to IaaS and it refers to a technique to

※ This paper was supported by wonkwang university in 2010.
Manuscript received August 9, 2010; accepted September 15, 2010.
Corresponding Author: Young-Sik Jeong
* Dept. of Computer Engineering, Wonkwang University, 344-2 Shinyoung-Dong, Iksan, Korea ({sjbaek, ssun, shyang, ehsong, ysjeong}@wku.ac.kr)

achieve physical and logical integration for the communicability increase of several servers.

Server virtualization technology as a new paradigm provides more efficiency than existing server systems in that it provides a decrease in server number, a decrease in server storage space, the decrease of server maintenance labour, a decrease in sever power maintenance, etc.. However, server virtualization generates other costs, namely, the introduction of a new server in keeping with the concept of operating several guest OS's with a single server and the fact that this must be a better performing server in comparison with the existing server(s). EMC (VMware) [3], Microsoft (Virtual Server 2005), SWsoft (Virtuozzo), Sun (Solaris Container), XenSource (Xen Enterprise3) [4], etc., which are server virtualization systems, are based on the transition from the existing server to a higher performance server and its subsequent integration. Many costs are generated by the introduction of this type of high performance server.

This study builds a virtualization server system using the existing server and grid service technology in order to resolve the expense increase issues generated by the introduction of this type of server. It is to provide performance enhancement and power efficiency using a sleep server that is a subsidized server and is based on a grid infrastructure. As shown in Fig. 1, the existing servers are partitioned into Meta server groups and the sleep server group is based on this grid infrastructure. The meta server is grouped according to a method in which each existing server is mapped onto a virtual node or virtual server. The sleep server is grouped according to a method in which each existing server is mapped onto a sleep node. These server configurations are scheduled at a virtual server when a calculation grid is used or a dispersed network become necessary. This study designs and realizes a Grid Infrastructure Virtualization server (GIV) which performs functions such as the scheduling of tasks and constituents and/or the monitoring of a task list, statistical treatments, etc.

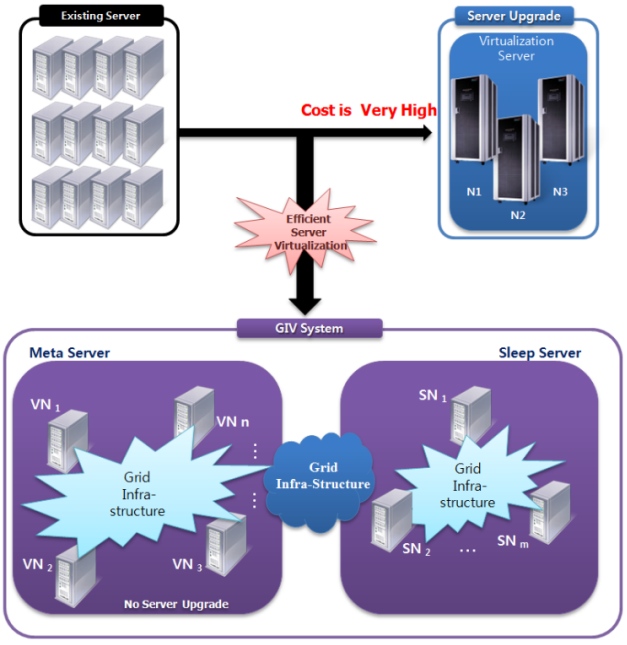

Fig. 1. Overview of GIV System

2. RELATED WORK

2.1 Server Virtualization

Server virtualization refers to the technology to physically operate several heterogeneous OS's on a single system. It is a technology used to divide a physical element into several logical elements or to integrate several physical elements into one logical element for their collective management. Server virtualization offers the benefit of decreasing system maintenance expenses by managing several physical or logical elements with a single device.

Methods for server virtualization are largely divided into H/W Partitioning, Logical Partitioning and SW Partitioning [2].

H/W Partitioning refers to a partitioning technique which completely separates elements into hardware units, that is, system board units. This technique provides complete performance and safety, but lowers system flexibility; for instance, sharing resources between partitions might be disallowed. Logical Partitioning refers to a partitioning technique which separates elements into micro code (firmware) units and supports the construction of more partitions (e.g. the process and the I/O partition are shared) in comparison with H/W Partitioning and accordingly allows a more efficient use of resources.

S/W Partitioning performs the partitioning task in software code units and is recognized as the most flexible form of partitioning as it shares resources such as partition, process, memory, I/O, disk, etc. However, it needs a Code layer (OS, Hypervisor) for partition management which presents a drawback in that it affects the entire system performance when any software problem occurs.

This study uses S/W Partitioning by Hypervisor with high flexibility and partitioning performance for the most efficient resource management. Hypervisor provides its abstraction and isolation functions to each virtual machine of the host system through CPU interrupt, state management, etc. Moreover, it has a software layer with an abstract concept utilized by its physical hardware which efficiently shares and distributes the physical server resources.

Table 1. Server partitioning for virtualization

	Partition	**Safety**	**flexibility**	**Performance**
H/W Partitioning	Low	High	Low	High
Logical Partitioning	Medium	Medium	Medium	Medium
Software Partitioning	High	Low	High	Low

2.2 Grid Infrastructure

Grid computing refers to performing computation intensive jobs and/or data intensive jobs by connecting every computing device such as PCs, servers, PDAs, etc. to one network [5]. To be precise, it is about the concept of collecting dispersed computing resources by a high speed network such as FDDI or internet (web) and utilizing them for massive task performance. Generally, since it is rare to use 100% of a CPU while performing computer tasks, a certain amount of idleness happens all the time. Grid computing is a technique to share the idle resources within a Grid Service Infrastructure and increase the performance speed focusing on a certain task [6].

Scheduling techniques for Grid computing are largely divided into Centralized Scheduling and De-Centralized Scheduling based on the ways of treating the tasks. In Centralized Schedul-

ing as shown in Fig. 2, a Meta scheduler manages the states of all nodes within a Gird computing environment. Since, in this method, a Meta scheduler manages all idle resources; it provides advantages such as ease of task management and system node configuration. However, since the Meta scheduler manages all information, a bottle neck phenomenon can occur in the presence of network problems [7].

In the De-Centralized Scheduling system of Fig. 3, multiple Meta schedulers exist to manage the schedule of each node. Each node performs scheduling for each node and grants a task to the node with the largest amount of idle resources. The advantages of this scheduling system are that it supports independent scheduling of each node and network interference does not affect the entire system.

However, it is hard to provide an optimized Grid computing environment due to the synchronization issue and idle resource management [7]. This study develops a technology by using De-Centralized Scheduling to achieve flexible job scheduling via the independent scheduling of each Meta node which decreases incidents of the bottleneck phenomenon.

The points of expenses and utilization of the existing server were heretofore frequently overlooked due to focusing only on the technology of server virtualization itself.

In this light, this study attempts to provide an efficient virtual computing environment by applying Resource/Task Scheduling techniques based on Grid Infrastructure and by utilizing efficient server power maintenance techniques.

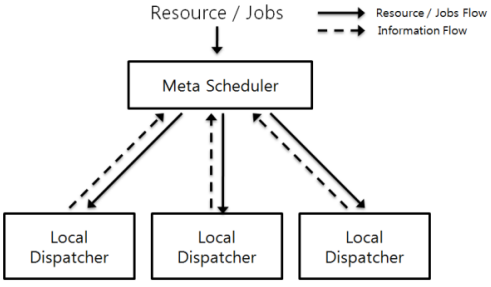

Fig. 2. Centralized Scheduling

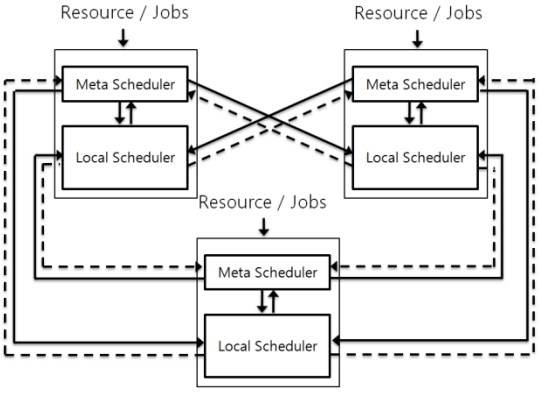

Fig. 3. De-Centralized Scheduling

3. EFFICIENT SERVER VIRTUALIZATION USING GRIDS

The existing systems enhanced power efficiency by improving server performance, configuring the virtualization server and then reducing the number of servers. This study suggests an alternative system structure that enhances power efficiency and performance by suing ACPI (Advanced Configuration & Power Interface) [8], which is a power management standard, and GRID Infrastructure.

The basic system structure is based on the following mathematical modeling.

$$f : \mathrm{ES} \longrightarrow < M, S >$$
$$\mathrm{es}_i \longmapsto < \mathrm{VN_j}, \mathrm{SN_k} >$$

Where, $\mathrm{es_i}$: exist server
$\mathrm{VN_j}$: meta server
$\mathrm{SN_k}$: sleep server
$i, j, k \in Z$

Function f is a one-to-one function having the existing servers as its domain and tuples of the Meta server and sleep server as its range. <M, S> are defined by Meta server and sleep sever tuples. M is a meta server, meaning the set of a virtual node, $M=\{VN_i \mid i \in Z\}$ and S is a sleep server, expressing the set of sleep nodes, $S=\{SN_i \mid i \in Z\}$. The basic rules for mapping are as follows:

Rule 1 : Cardinality $|M| \geq |S|$
Rule 2 : $^{\forall}es_i$, $f(es_i)= <VN_j, \emptyset>$ or $<\emptyset, SN_k>$, $i,j,k \in Z$
Rule 3 : For each $^{\forall}SN_i$, $2^0 \leq |VN_j| \leq 2^4$

Rule 1 means that cardinality (|M|) of the set of a virtual node is always the same as or larger than cardinality (|S|) of the set of a sleep node.

Rule 2 is that <M, S> which is always a range element on the mapping, corresponds to whichever one of them that has an empty element value. A random element S_i of ES always gets mapped as one among $< \emptyset, SN_i>$. Then, VN_i ($i \in Z$) is a random element belonging to M at the <M, S> tuple and SN_i ($i \in Z$) means mapping onto a random element belonging to S at the <M, S> tuple. That is to say, it means that a random exiting server must be mapped onto one Meta server or one sleep server.

Rule 3 means that the value on the number of Meta servers per sleep server is decided between $2^0 \sim 2^4$, when the existing servers are mapped onto the Meta server or the sleep server.

This section described an example of a GIV system configuration of parameters such that ES has eight existing servers and the entire system of <M, S> was configured with six VN servers for M tuple, two SN servers for S tuple by function f. Also, the Grid Scheduling of this virtual server system performs scheduling with the Meta data of whether idle resources exist or not (MetaNodeList), whether virtualization server tasks exist or not (WorkFlowInfo) and whether sleep server activity exists or not, all in order to share the CPU's load and disperse the network as a calculation grid, with the aim of providing smooth web service.

3.1 Grid Scheduling

Set M of VN's and Set S of SN's at <M, S> are configured as Grid infrastructure and the rela-

```
MetaNodeWorkFlow(MNL, SNL, WFI)

INPUTE : MetaNodeList MNL, SleepNodeList SNL,
         WorkFlowInfo WFI

While (  True  )
    if (  this.Overflow == True  )
        if ( MNL.GetResources ( ) == True  )
            NodeConnection ( MNL, WFI  )
        End if
        else if ( SNL.GetSleep ( ) == True  )
            NodeConnection ( SNL, WFI  )
            SNL.setSleep (  False  )
        End else if
    End if
    else
        LocalNodeWork( )
    End else
End while
```

Fig. 4. MetaNodeWorkFlow Scheduling

tionship between the two sets are also configured as Grid. The scheduling and network, etc. between Set M and S, respectively of VN_i ($i \in Z$) and SN_i ($i \in Z$) are the subjects of Grid Scheduling.

At the VN server, the computer resources (CPU, Network) can be exhausted due to a user explosion of web service. This exhaustion can be caused by a down server and does not provide the users with a QoS. In this study, the Scheduling of Grid Infrastructure starts when 70% of the VN server's resources (work overflow) are used. Scheduling is performed according to the results of searching whether the idle resources able to perform the tasks of adjacent nodes exist or not, and the corresponding procedure is as shown in Fig. 4. The node selecting scheduling performs node searching using the inputs of the VN server information (Metanodelist : MNL), SN server information (SleepNodelist : SNL) and task information (WorkflowInfo : WFI).

After an overflow is generated at VN if any idle resources are found when searching the adjacent servers of the adjacent VN_i then a data transfer occurs from Server, MNL and WFI by connecting to the corresponding node. However, if no idle resources are found among the servers connected to VN_i, it transforms the waiting SN_i in the sleep node among SN_i and performs the NodeConnection.

3.2 Efficiency Power Scheme

VN_i belonging to the Meta server do not have any problems in performance although they use the existing servers. However, the limits of the existing server's performance are occasionally reached when a server overload occurs due to a user explosion.

To overcome this limit of the server, SN_i exists as a subsidized server. However, if SN, the subsidized server is on standby in an idle state, it causes a problem of power use equivalent to adding an extra server. To resolve this problem, servers of $S=\{SN_i \mid i \in Z\}$stand by always in sleep mode.

$S=\{SN_i \mid i \in Z\}$follows ACPI, the standards for computer power management to achieve this sleep mode. ACPI is defined as being divided into two groups of the power management desktop, namely, the Global System and the external device. The external device comprises four power states from D0 to D, and the power management of the computer desktop is called the Global System State and is divided into four steps from G0 to G3. If looking at the Table 1, which compares the ACPI modes on actual computers, the sleep mode uses 0.7~1w more power in comparison with the OFF mode [9].

Table 2. ACPI Sleep Mode

Mode	IDLE	Activity	Sleep	Max Sleep	Power Off
PC	73.0 w	77.0 w	3.97 w	3.27w	3.27 w
Monitor	32.5 w	33.5 w	1.18 w	1.18 w	1.18 w
Total	105.5 w	110.5 w	5.12 w	4.45 w	4.45 w

Table 3. Global System State (G0~G3)

G state	explanation	
G0 (S0)	Working	
G1	Sleeping subdivides into the four states S1 through S4	
	G1 state	explanation
	S0 (G0)	Working
	S1	All processor caches are flushed, and the CPU(s) stop executing instructions. Power to the CPU(s) and RAM is maintained; devices that do not indicate they must remain on may be powered down.
	S2	CPU powered off
	S3	Commonly referred to as Standby, Sleep, or Suspend to RAM. RAM remains powered
	S4	Hibernation or Suspend to Disk. All content of main memory is saved to non-volatile memory such as a hard drive, and is powered down.
	S5 (G2)	Powered off
G2(S5)	G2 is almost the same as G3 *Mechanical Off*, but some components remain powered so the computer can "wake" from input from the keyboard, clock, modem, LAN, or USB device.	
G3	The computer's power consumption approaches close to zero, to the point that the power cord can be removed and the system is safe for dis-assembly (typically, only the real-time clock is running off its own small battery).	

The sleep server uses G1 among various modes of the Global System State. In this study, standby mode is achieved using the S3 sleep mode which can reduce more than 20 times the power use in comparison with the idle state

4. DESIGN OF GIV

The entire system structure of the GIV including the Meta Server Node Frame, View Frame and Sleep Serve Node Frame are as shown in Fig. 5.

VN and SN are generated and added at the View Frame. Then, the View Manager transfers

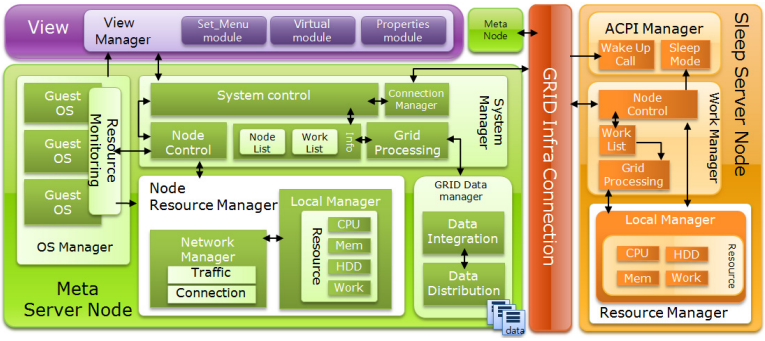

Fig. 5. GIV Architecture

the information to the System Manager of the VN (Meta Node). When task operation and overflow occur, the System Manager performs Grid Scheduling and operates the system.

4.1 GUI View Frame

View Frame is a GUI Application for user convenience. View Manager comprises three modules; the SetMenu Module, Virtual Module and Properties Module, and each module illustrates the management of VN addition/deletion and the attribute information of the node (Guest OS Resource, Work flow). The user can add VN or SN onto the GIV system at the SetMenu Module of the View Manager, and can start or stop the system monitoring. The Virtual Module visually provides the information on the server addition status or Grid utilization status, etc. in order to help the user's understandings.

The Properties Module visually provides the attribute information of these additional servers.

4.2 Meta Server Node Frame

VN (Meta Server node) consists of System Manager, OS Manager, Resource manager and Grid Data manager. The System Manager manages the local and the global grid systems through System Control, as one of the core modules. Also, it manages guest OS resources through OS Manager and starts Resource/Task Scheduling with this information.

Resource/Task Scheduling performs Grid Processing after checking the idle resources among various ${}^{\forall}\mathrm{VN}_i\,(\in \mathrm{M},\ i \in Z\}$ according to Resource/Task Scheduling if an overflow occurs while treating the task.

4.3 Sleep Server Node Frame

$\mathrm{S}=\{\mathrm{SN}_i \mid i \in Z\}$ consists of the ACPI Manager which manages the power and the Work Manager which manages the resources. Regarding power management, it is always maintained in the sleep mode. This sleep mode changes to normal mode from the Wake Up Call of the ACPI Manager when Grid Data generate at $\mathrm{M}=\{\mathrm{VN}_i \mid i \in Z\}$. It transfers Grid Data and Work List information to the System Manager and Work Manager of the Sleep Server Node, and the sleep mode processes the data through Grid Processing.

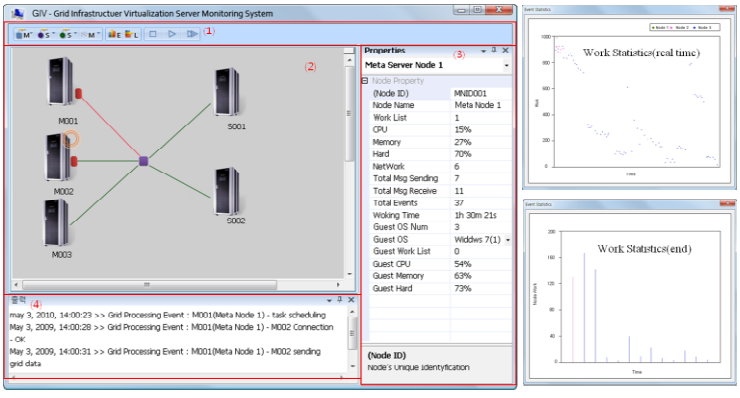

Fig. 6. Visualization of GIV

5. IMPLEMENTATION

GIV Monitoring consists of a menu bar and the tool collection (①), Visual Window (②), Server Attribute Window (③) and Event Printing Window (④).

The Visual Window provides the server configuration status and the activity status information. The state of Grid Work Flow is marked with colors and is discernable based on the color change. It is possible to add a virtual server and a sleep server by using the tool collection and the right hand button of the mouse. A way to connect these servers is adding IP after running the program. The Visual Window visualizes the system configuration state and work flow state with the color change of the lines. The Server Attribute Window presents the server attributes, the current state (work list, CPU, memory, hard drive, network, msg., event, working time and guest OS information), etc. and the system shows it in the Event Printing Window upon any event generation. At the tool collection menu, the system monitoring starts and stops. Statistical graphs on the system (End Time, Real Time) are provided.

6. CONCLUSION

The system suggested in this study enhances performance and power efficiency in comparison with the existing server systems, by utilizing Grid Service Technology and using the idle resources of the virtualization server and the sleep server. In other words, it was developed by using the existing servers instead of introducing a new server and by integrating Grid Infrastructure Technology. Furthermore, it was possible to achieve a reduction in power usage by adding a subsidized server and enhancing the performance. Of particular note is the fact that, now it is possible to utilize the current idle resource state and task schedule of the server by using a monitoring program, and to estimate the expense of power usage by monitoring the power usage.

Further researches on security for Grid Infrastructures [10], the solubility increase of idle resources within the architecture and the algorithms/scheduling for efficient task distribution are intended in the future.

REFERENCE

[1] Jackie Fenn, Ken McGee, Nark Raskino, Kathy Harris, and David W. Cearley, “Key Issues for Emerging Trends,” Gartner Research, April, 2007.

[2] Paul Mason and Dan Kusnetzky, “Server Provisioning, Virtualization, and the On-demand Model of Computing: Addressing Market Confusion,” IDC, June, 2003.

[3] VMware, Inc. “Understanding Full Virtualization, Paravirtualization, and Hardware Assist,” http://www.vmare.com.2007.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the Art of Virtualization,”Proc. of the 19th ACM Symposium on SOSP, 2003.

[5] I. Foster, et aI., “Grid Services for Distributed System Integration,” Computer, Vol.35, No.6, 2002, pp.37-46.

[6] K. Krauter, et aI., “A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing,” Software Practice and Experience Journal, Vol.32, No.2, February, 2002, pp.135-164.

[7] V. Hamscher, et. Al., “Evalution of Job-Scheduling Strategies for Grid Computing”, The 1st IEEE/ACM international Workshop on Grid Computing (Grid 2000) at the 7th International Conference on High Performance Computing (HiPC-2000), LNCS 1971, 2000, pp.191-202.

[8] “Advanced Configuration and Power Interface Specification” http://www.acpi.info/DOWNLOADS/

ACPIspec30.pdf
[9] Kwon Won-Ok, Kim Sung-Un "PC Power Management" http://www.itfind.or.kr/WZIN/jugidong/1344/ file35290-134401.pdf
[10] J. K. Jones, G. W. Romney, "Honeyuets: an educational resource for IT security," Proc. of the 5th conference on Information technology education, 2004.

Sung-Jin Baek

He received a B.S. degree in computer engineering from Wonkwnag University, Iksan, Korea in 2008. Since 2008, he has been with the Department of Computer Engineering, Wonkwang University, Iksan, Korea, where he studies for a master's degree, working in the areas of Grid Computing and Semantic Grid, distributed and mobile computing, USN middleware.

Sun-Mi Park

She received a B.S. degree in computer engineering from Wonkwnag University, Iksan, Korea in 2006. Since 2006, she has been with the Department of Computer Engineering, Wonkwang University, Iksan, Korea, where she studies for a master's degree, working in the areas of Grid Computing and Semantic Grid, distributed and mobile computing, USN middleware.

Su-Hyun Yang

She received a B.S. degree in computer engineering from Wonkwnag University, Iksan, Korea in 2010. Since 2010, she has been with the Department of Computer Engineering, Wonkwang University, Iksan, Korea, where she studies for a master's degree, working in the areas of Grid Computing and Semantic Grid, distributed and mobile computing, USN middleware.

Eun-Ha Song

She received a B.S. degree in statistics and an M.S., and Ph.D. degree in computer engineering from Wonkwnag University, Iksan, Korea in 1997, 2000 and 2006 respectively. Since 2007, she has been with the Department of Computer Engineering, Wonkwang University, Iksan, Korea, where she is a full-time lecturer, working in the areas of Grid Computing and Semantic Grids, distributed and mobile computing, USN middleware.

Young-Sik Jeong

He received a B.S. degree in mathematics and an M.S., and Ph.D. degree in computer science and engineering from Korea University, Seoul, Korea in 1987, 1989, 1993, respectively. Since 1993, he has been with the Department of Computer Engineering, Wonkwang University, Iksan, Korea, where he is a professor, working in the areas of Grid Computing and Semantic Grids, distributed and mobile computing, USN middleware.