http://dx.doi.org/10.5392/IJoC.2016.12.2.006

Privacy Enhanced Data Security Mechanism in a Large-Scale Distributed Computing System for HTC and MTC

Seungwoo Rho
National Institute of Supercomputing and Networking
KISTI, 335 Gwahangno, Yuseong, Daejeon, 305-806, South Korea

Sangbae Park
National Institute of Supercomputing and Networking
KISTI, 335 Gwahangno, Yuseong, Daejeon, 305-806, South Korea

Soonwook Hwang
National Institute of Supercomputing and Networking
KISTI, 335 Gwahangno, Yuseong, Daejeon, 305-806, South Korea

ABSTRACT

We developed a pilot-job based large-scale distributed computing system to support HTC and MTC, called HTCaaS (High-Throughput Computing as a Service), which helps scientists solve large-scale scientific problems in areas such as pharmaceutical domains, high-energy physics, nuclear physics and bio science. Since most of these problems involve critical data that affect the national economy and activate basic industries, data privacy is a very important issue. In this paper, we implement a privacy enhanced data security mechanism to support HTC and MTC in a large-scale distributed computing system and show how this technique affects performance in our system. With this mechanism, users can securely store data in our system.

Key words*: HTC, MTC, Privacy, Security, Pilot Job, HTCaaS, Distributed Computing.*

1. INTRODUCTION

Recently, there has been a further increase in the utilization of geographically distributed and heterogeneous computing resources. Overall distributed computing paradigms can be categorized into three types, namely, HTC, HPC and MTC [1]. High-Throughput Computing (HTC) supports the use of many computing resources over long periods of time to perform computational tasks and High Performance Computing (HPC) processes for communication-intensive tasks on top of dedicated clusters of workstations or supercomputers. Many-task computing (MTC) in computational science is an approach to parallel computing that aims to bridge the gap between two computing paradigms, HTC and HPC. MTC uses many computing resources over short periods of time to accomplish many computational tasks, including both dependent and independent tasks.

Service infrastructures such as Grid, Cloud and Supercomputer are mainly operated in remote distributed areas to support many resources. To conveniently support such infrastructures to researchers, we developed a pilot-job based large-scale distributed computing system, called HTCaaS [2] (High-Throughput Computing as a Service), which helps scientists solve large-scale scientific problems in areas such as pharmaceutical domains, high-energy physics, nuclear physics and bio science. However, using outside distributed computing resources can give rise to security problems related to the data privacy of scientists. This is the reason many prefer using only local cluster resources such as PBS [3], LoadLeveler [4] and HTCondor [5]. Also, because many of these cases involve critical data that may affect the national economy and activate basic industries, data privacy is all the more important. The contribution of this paper is to implement a privacy enhanced data security mechanism in a large-scale distributed computing system, and we show how this technique affects performance in HTCaaS.

The remainder of this paper is organized as follows. Section 2 discusses both previous studies and related studies. In Section 3, we introduce our system HTCaaS, implement its data security mechanism and present the obtained performance

This is an excellent paper selected from the papers presented at ICCC 2015.
** Corresponding author, Email: seungwoo0926@kisti.re.kr*
Manuscript received Apr. 06, 2016; revised May. 25, 2016; accepted May. 30, 2016

results with explanations. In Section 4, we summarize the paper and present directions for further research.

2. RELATED WORKS

In this section, we will briefly introduce the pilot-job [6] system and discuss some its security issues. We also introduce MTC/HTC applications.

2.1 Pilot Job System and Security

Since pilot-job systems have been developed with a focus on improving availability and since they are based on multi-level scheduling, they have some shortcomings in regards to privacy. The pilot or agent is the entity that actually gets submitted and scheduled onto a resource using the local resource management system, such as the portable batch system. The pilot collects local information, manages the resources allocated to it and exchanges data (such as resource information, pilot states). This method solves some of the problems that occurred in other existing schedulers, such as the long waiting time of a queue, periodic resource allocation and de-allocation, errors arising from user tasks using the push method, and the heavy load caused by periodic status monitoring of jobs, and also has many other advantages, including the ability to dynamically inspect and prepare the application environment. Examples of such pilot-based multi-level schedulers include Falkon [7], BigJob [8], MyCluster [9], DIRAC [10], etc. These schedulers all feature infrastructures based on different targets.

This method of scheduling, however, does not depend on the existing security method with first level scheduling, and therefore a separate security policy must be implemented to safeguard the data handled by the pilots. In this paper, we present a mechanism that applies a pilot security policy to HTCaaS and demonstrate its performance based on application results.

2.2 HTC/MTC Applications

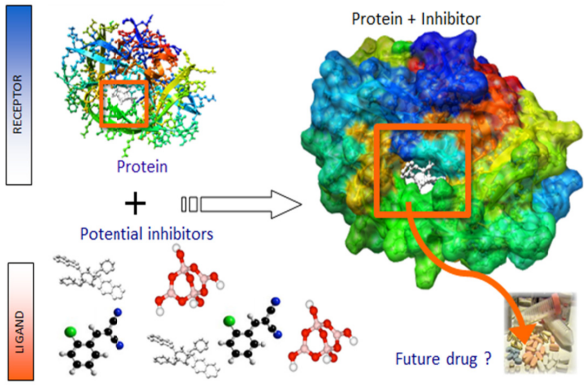

Fig. 1. Virtual screening using molecular mocking

2.1.1 Pharmaceutical Domain: Pharmaceutical domain applications screen KEGG [11] (Ligand Database) compounds and drugs against important metabolic protein targets using DOCK6 [12]. It needs more than one billion computations with a large variance of execution times from seconds to hours (average 10 minutes) and there is little to no prior knowledge about execution time. In addition, it involves significant I/O for each computation as the compounds are typically stored in a database (i.e. 10s to 100s of MB large) and must be read completely per computation.

In this paper, we focus on virtual screening using molecular docking (autodock vina). Autodock vina is a suite of automated docking tools which perform the docking of ligands to a set of target proteins to discover new drugs for several serious diseases such as SARS or Malaria. HTCaaS can successfully complete the entire process of virtual screening by dynamically acquiring and managing computing resources.

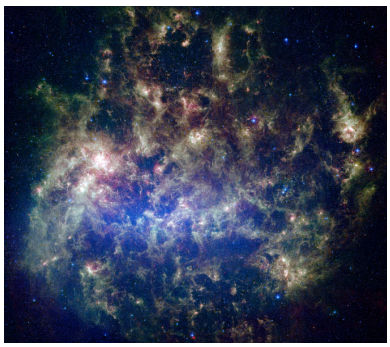

Fig. 2. MONTAGE program

2.2.2 Astronomy Applications: MONTAGE is a national virtual observatory project which stitches tiles of images of the sky from various sky surveys (e.g. SDSS, etc.) into a photorealistic single image. The execution times per task are in the range of 100ms to 10s of seconds, and it involves multiple input images and at least one image output.

It is both computing-intensive and data-intensive and its scalability is limited when run under HTC or HPC.

3. HTCAAS PRIVACY MODE

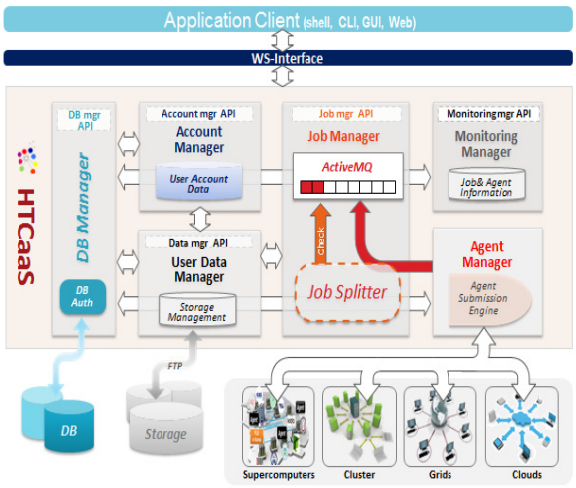

Fig. 3. HTCaaS system architecture

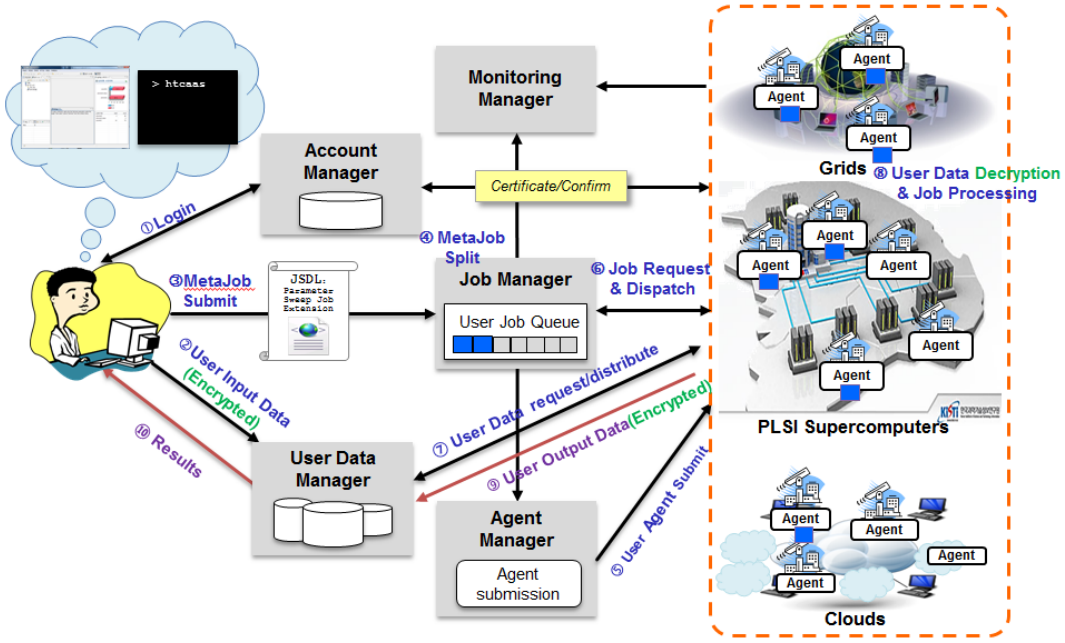

Fig. 4. The flow diagram of HTCaaS privacy mode

In this section, we briefly describe our system called HTCaaS and give our work on applying security technology to user data and experimental results that can address the performance issue to support security of many tasks.

3.1 High-Throughput Computing as a Service (HTCaaS)

To support HTC/MTC applications, we considered several principles (ease of use, efficient task dispatching, adaptiveness, fairness, reliability, resource integration). We designed and implemented HTCaaS to provide researchers with ease of exploring large-scale and complex scientific problems by effectively leveraging a hybrid and distributed computing infrastructure. Fig. 3 presents an overview of the system structure and flow of HTCaaS. HTCaaS consists of 6 independent modules including DB Manager (system meta data management), Account Manager (user data management and integrated authentication / authorization), Job Manager (job life cycle management), Monitoring Manager (job and agent monitoring), User Data Manager (application data and execution file management), and Agent Manager (agent management). These communicate with one another via web service. Furthermore, each module provides an open web service to improve the ease of using HTCaaS from an external site.

HTCaaS exploits a synthesis of well-known techniques and its own intelligent scheduling algorithm to effectively support multiple users independently submitting large numbers of tasks to a collection of computing resources and has several main features for supporting HTC/MTC efficiently.

First, we use OGF JSDL [13] and parameter sweep job extension as a job description language. Parameter sweep job extension support meta-job based automatic job split and submission technique. This enables researchers to easily submit and execute hundreds of thousands of jobs at once, simply expressed by a single JSDL script.

Second, we adopted agent-based (generally called pilot) multi-level scheduling. Each agent actively pulls a user task (job) at one time, processes it and records the statistics independently. In other words, HTCaaS intelligently creates a dedicated resource pool on the fly across heterogeneous computing resources. Also, HTCaaS maintains separate job queues and agents per user to reduce the complexities of accounting and scheduling on a general infrastructure such as PLSI [14] (Partnership & Leadership for the nationwide Supercomputing Infrastructure), which provides researchers with an integrated view of geographically distributed supercomputing infrastructures in Korea. On the other hand, HTCaaS keeps a single job queue and global agents for grid infrastructure.

Third, HTCaaS implements user-level scheduling and dynamic load balancing mechanisms by adjusting the number of agents allocated to a user. Because the overall load distribution can change due to the heterogeneity in task execution times and resource capabilities, we need to consistently monitor changes in the overall load distribution and effectively adapt to the changing load. To provide a dynamic load balancing mechanism, we developed a dynamic fair resource sharing algorithm. It divides all available computing resources fairly across all demanding users in the system (when the system is heavily loaded) and exploits dynamic adjustment of acquired resources as free computing resources become available (as the overall system becomes lightly loaded).

3.2 The Implementation of Data Security Mechanism

Because our system globally uses many hybrid schedulers like PBS, LoadLeveler, SunGridEngine, Openstack, HTCondor and gLite at the same time, it is very important to protect the privacy of the research data. For example, in the pharmaceutical and high energy physics fields, the data must be protected very carefully. To satisfy such a requirement, we

implemented a security mechanism on our system. Fig. 4 describes the overall security mechanism in HTCaaS. This mechanism largely consists of 3 main steps and 7 sub steps.

(Preparatory Step)

Step 1. The user logs into the server and uploads encrypted data (Stages 1 and 2).

Step 2. The user writes two files. One is a shell executable script file (Fig. 5) that can decrypt the user's encrypted input data and encrypt output data. The other is a JSDL file (Fig. 6) that can run the shell executable script file (Stage 3).

Step 3. The user submits the JSDL file to a server (Stage 3).

```
ligand=$2
dir=$3

PAIR=$ligand

export AUTODOCK_UTI=$PWD
export LD_LIBRARY_PATH=$PWD:$LD_LIBRARY_PATH
ulimit -s unlimited

chmod +x ./vina

openssl enc -aes-128-cbc -d -in $ligand.pdbqt_enc -out $ligand.pdbqt -pass pass:shtmddn
openssl enc -aes-128-cbc -d -in $protein.pdbqt_enc -out $protein.pdbqt -pass pass:shtmddn


echo "Docking Start"
./vina --receptor $protein.pdbqt --ligand $ligand.pdbqt
--out $PAIR.out --log $PAIR.log --center_x 10.18 --center_y 8.3 --center_z 9.94 --size_x 20
--size_y 20 --size_z 20 --cpu 1 --seed 1
```

Fig. 5. The shell executable script file (autodock_vina_enc.sh)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<JobDefinition xmlns="http://schemas.ggf.org/jsdl/2005/11/jsdl" xmlns:ns2="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
xmlns:ns3="http://schemas.ogf.org/jsdl/2009/03/sweep">
    <JobDescription>
        <Application>
            <ApplicationName>autodock3</ApplicationName>
            <ns2:POSIXApplication>
                <ns2:Executable>autodock_vina_enc.sh</ns2:Executable>
                <ns2:Argument>TARGET</ns2:Argument>
                <ns2:Argument>LIGAND</ns2:Argument>
            </ns2:POSIXApplication>
        </Application>

        <DataStaging>
            <FileName>LIGAND.pdbqt_enc</FileName>
            <CreationFlag>overwrite</CreationFlag>
            <Source>
                <URI>/pwork01/p258rsw/privacy/separated_pdbqt/privacy/100_enc/LIGAND.pdbqt_enc</URI>
            </Source>
        </DataStaging>

        <DataStaging>
            <FileName>TARGET.pdbqt_enc</FileName>
            <Source>
                <URI>/pwork01/p258rsw/privacy/protein/TARGET.pdbqt_enc</URI>
            </Source>
        </DataStaging>

        <DataStaging>
            <FileName>autodock_vina_enc.sh</FileName>
            <Source>
                <URI>/pwork01/p258rsw/privacy/autodock_vina_enc.sh</URI>
            </Source>
        </DataStaging>

        <DataStaging>
            <FileName>vina</FileName>
            <Source>
                <URI>/pwork01/p258rsw/privacy/vina</URI>
            </Source>
        </DataStaging>

        <DataStaging>
            <FileName>LIGAND.out_enc</FileName>
            <Target>
                <URI>/pwork01/p258rsw/privacy/output/privacy/100_enc/LIGAND.out_enc</URI>
            </Target>
        </DataStaging>

        <DataStaging>
            <FileName>LIGAND.log</FileName>
            <Target>
                <URI>/pwork01/p258rsw/privacy/output/privacy/100_enc/LIGAND.log</URI>
            </Target>
        </DataStaging>

    </JobDescription>
```

Fig. 6. The JSDL file with 100 ligands

```
    <ns3:Sweep>
        <ns3:Assignment>
            <ns3:DocumentNode>
                <ns3:Match>TARGET</ns3:Match>
            </ns3:DocumentNode>
            <ns3:Directory filenameonly="true">
                <ns3:Name>/pwork01/p258rsw/privacy/protein</ns3:Name>
            </ns3:Directory>
        </ns3:Assignment>
        <ns3:Sweep>
            <ns3:Assignment>
                <ns3:DocumentNode>
                    <ns3:Match>LIGAND</ns3:Match>
                </ns3:DocumentNode>
                <ns3:Directory filenameonly="true">
                    <ns3:Name>/pwork01/p258rsw/privacy/separated_pdbqt/privacy/100_enc</ns3:Name>
                </ns3:Directory>
            </ns3:Assignment>
        </ns3:Sweep>
    </ns3:Sweep>
</JobDefinition>
```

Fig. 7. The Sweep part of JSDL file

(Computational Step)

Step 4. The server automatically splits the meta-job of the user into many sub-jobs according to the JSDL file and puts them into the user job queue (Stage 4).

Step 5. The server deploys user agents on computational nodes such as Grids, PLSI, and Cloud (Stage 5).

Step 6. Each User Agent pulls the user job and the encrypted user data. After processing the jobs, it encrypts output data and uploads the data to the server (Stages 6, 7, 8, and 9).

(Final Step)

Step 7. The user accesses the server and downloads & decrypts the results (Stage 10).

This security mechanism is almost equal to the existing non-security system except that the user and the agents deal with encrypted data. We only need to edit the JSDL and executable files so that the agent processes the encrypted data. To compare how this cryptographical method affects our system, we conducted four kinds of Autodock Vina [15] experiments. In this experiment, we used the testbed with 3.00 GHz, 40 cores as a server node, 104 cores as compute nodes and a security method with a 128 bit AES algorithm as an encryption algorithm.

3.3 Experiments

In this experiment, we repeated the docking test several times and measured the total makespan when the number of ligands was set at 4 variations, respectively 100, 200, 400, and 800. To confirm the reduction in performance time during file I/O and calculations, we performed an additional experiment with ramdisk, while using the NFS, which is the default file system for HTCaaS, as the reference for comparison.

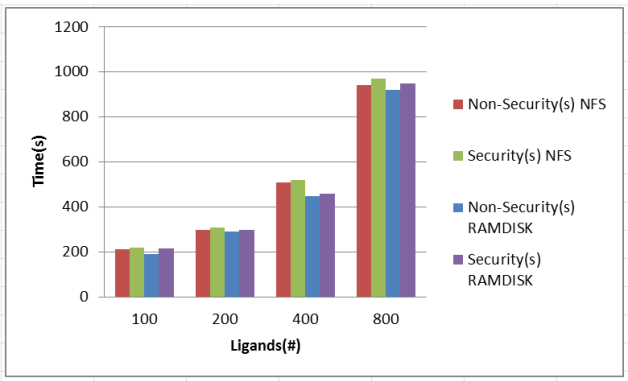

Fig. 8. Autodock Vina Experiment Results

First of all, Fig. 8 shows that there are no big differences between two methods (security and non-security). This means that the overhead for cryptographical operations is very small. In the cases where the number of ligands was respectively 200 and 800, the ramdisk method show similar performance compared with the nfs method. This is because Autodock Vina is a virtual program for performing molecular docking. This means that Autodock Vina can have a slightly different

makespan whenever the same docking process is repeated, but the margin of discrepancy is very small.

Comparing the cases in which nfs and ramdisk were respectively used, we observed that there was no notable difference up to all the case, but when there were only 400 ligands, the makespan reduced slightly in the case that the ramdisk was used. In other words, the use of ramdisk remarkably does not produce an effect on the performance. This means that because autodock vina tasks are almost compute-intensive, all the performance is similar compared to cases in which the NFS is used.

4. CONCLUSION AND FUTURE RESEARCH

In this paper, we introduced HTCaaS that supports the exploration of large-scale and complex HTC/MTC problems employing the concept of meta-job (with easy-to-use client tools) and a fault-tolerant agent-based multi-level scheduling mechanism (with ease of use, efficient task dispatching and reliability). Especially we implemented the privacy enhanced data security mechanism in HTCaaS to enable scientists to keep their data securely. We also presented our experiment results comparing the security and non-security methods in an example from the pharmaceutical domain.

By applying this security concept, HTCaaS is able to easily support enhanced privacy for data-sensitive users. Our experimental result also showed that there is little overhead for the cryptographical operations

In the future, we will support more complex workloads consisting of HTC, MTC and HPC tasks and improve the scalability of HTCaaS.

ACKNOWLEDGEMENT

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government (MSIP) (No.R0190-15-2012, High Performance Big Data Analytics Platform Performance Acceleration Technologies Development) and Korea Ministry of Environment (MOE) as Public Technology Program based on Environmental Policy (Grant No.: 2014000210004).

REFERENCES

[1] I. Raicu, I. Foster, and Y. Zhao, “Many-Task Computing for Grids and Supercomputers,” Proc. Workshop on Many-Task Computing on Grids and Supercomputers, 2008, pp. 1-11.

[2] Jik-Soo Kim, Seungwoo Rho, Seoyoung Kim, Sangwan Kim, Seokkyoo Kim, and Soonwook Hwang, “HTCaaS: Leveraging Distributed Supercomputing Infrastructures for Large-Scale Scientific Computing,” Proc. Workshop on Many-Task Computing on Clouds, Grids, and Supercomputers, 2013.

[3] R. Henderson and D. Tweten, “Portable Batch System: External reference specification,” Technical report, NASA Ames Research Center, 1996.

[4] A. Prenneis, Jr, “LoadLeveler: Workload Management for Parallel and Distributed Computing Environments,” Proc. of Supercomputing Europe, 1996.

[5] HTCondor, http://research.cs wisc.edu/htcondor/htc.html, 2016.

[6] A. Luckow, M. Santcroos, O. Weider, A. Merzky, S. Maddineni, and S. Jha, “Towards a common model for pilot-jobs,” Proc. of The International ACM Symposium on High-Performance Parallel and Distributed Computing, 2012.

[7] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M.Wilde, “Falkon: a Fast and Light-weight tasK executiON framework,” Proc. ACM/IEEE conference on Supercomputing, 2007, pp. 1-12.

[8] A. Luckow, L. Lacinski, and S. Jha, “SAGA BigJob: An Extensible and Interoperable Pilot-Job Abstraction for Distributed Applications and Systems,” Proc. IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, pp. 135-144.

[9] E. Walker, J. P. Gardner, V. Litvin, and E. L. Turner, “Creating Personal Adaptive Clusters for Managing Scientific Jobs in a Distributed Computing Environment,” Proc. Challenges of Large Applications in Distributed Environments, 2006, pp. 95-103.

[10] A. Tsaregorodtsev, M. Bargiotti, N. Brook, A. C. Ramo, G. Castellani, P. Charpentier, C. Cioffi, J. Closier, R. G. Diaz, G. Kuznetsov, Y. Y. Li, R. Nandakumar, S. Paterson, R. Santinelli, A. C. Smith, M. S. Miguelez, and S. G. Jimenez, “DIRAC: a community grid solution,” Journal of Physics: Conference Series, 2008.

[11] KEGG, https://en.wikipedia.org/wiki/KEGG, 2016.

[12] Lang PT, Brozell SR, Mukherjee S, Pettersen EF, Meng EC, Thomas V, “DOCK 6: combining techniques to model RNA–small molecule complexes,” RNA, 2009.

[13] Open Grid Forum Job Submission Description Language, http://www.gridforum.org/ documents/GFD.56.pdf, 2016.

[14] Partnership & Leadership for the nationwide Supercomputing Infrastructure, http://www.plsi.or.kr/, 2016.

[15] O. Trott and A. J. Olson, “AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading,” Journal of Computational Chemistry, vol. 31, 2010, pp. 455-461.

Seungwoo Rho

He received the B.S., M.S in electrical and computer engineering from University of Seoul, Korea in 2009, 2011 respectively. Since then, he has been with the National Super Computing Center, Korea Institute of Science and Technology Information. His main research interests include supercomputer, cloud, distributed computing, and grid computing.

Sang-Bae Park

He received the B.S. in Mathematics from Seogang University and, M.S from POSTECH in 1993 and 1995. Since then, he has been with IDIS, SoftForum and KISTI. His main research interests include cryptography and information security, and application protection.

Soonwook Hwang

He received the B.S in Mathematics and the M.S in Computer Science and statistics from Seoul National University, Korea in 1990 and 1995 respectively. He also received the Ph.D. in Computer Science from University of Southern California in 2003. Since then, he has been with the National Institute of Informatics, Japan and the National Super Computing Center, Korea Institute of Science and Technology Information. His main research interests include grid, cloud, and distributed computing.