The Comprehensive Performance Rating for Hadoop Clusters on Cloud Computing Platform

Fong-Hao Liu, Ya-Ruei Liou, Hsiang-Fu Lo, Ko-Chin Chang, and Wei-Tsong Lee

***Abstract*—Virtualization platform solutions throughout the IT infrastructure are one important type of Green IT services in cloud data center. The Hadoop clusters composed of on-demand virtual infrastructure are used as an effective implementation of MapReduce for developing data intensive applications in cloud computing. Deploying Hadoop clusters on large numbers of data center virtual machines (VMs) can significantly increase its productivity and reduce both energy and resource consumption. However, the interference between the VMs is complicated and changing with the growth of data size. On the other hand, it would also decrease the performance of Map and Reduce tasks while using Hadoop clusters on virtual machines. In this paper, a Comprehensive Performance Rating (CPR) scheme is presented to probe the root causes of these problems, and solutions for VMs interference are introduced using data locality and excessive configuration parameters. Unlike previous solutions by customizing Hadoop native job scheduler, the proposed CPR scheme uses Hadoop configuration metrics revealed through Principal Component Analysis (PCA) method to guide the performance tuning work. The experimental data is resulted on a 20-node virtual cluster demonstration. The proposed CPR scheme performance is close to the measured execution time in different data size, cluster size and map tasks ratio.**

***Index Terms*—Hadoop, mapreduce, data locality, principal component analysis.**

I. Introduction

Cloud computing is currently a widely-known technology that is used as a service on demand. Cloud computing can combine with virtualization technology to create a Green computing infrastructure with high scalability, high energy efficiency and low resource consumption. It has changed the current trends in computer technology. The National Institute of Standards and Technology's (NIST) [1] has divided the taxonomies in cloud computing that characterizes several important aspects of cloud computing service and deployment models as shown in Table I.

PaaS is a service model that mainly focuses on flexibly providing consumers with its underlying cloud infrastructure, which includes network, servers, operating systems, and storage that control over the deployed applications. Hadoop is the open source that that plays an important role in constructing the cloud infrastructure. The reason behind this is Google use it as a key source in its MapReduce programming model. It is originally developed by Yahoo [2] for a reliable, scalable and well distributed parallel processing for big data sets across multiple hosts. The Hadoop core framework consists of a shared file system (HDFS), a set of distributed processing and an implementation of the MapReduce programming model. It is designed mainly focusing on running large files of data, therefore when encountering huge number of small files; it would run out of the system memory resources. Methods are investigated to improve the problem as listed in Table II [3]-[7]. In this paper, research is proposed to ameliorate the problem as well.

Manuscript received May 2, 2014; revised June 30, 2014. This work was supported by the National Science Council of Republic of China under grant NSC 102-2221-E-606-008.

Fong-Hao Liu, Ya-Ruei Liou, Hsiang-Fu Lo, and Ko-Chin Chang are with National Defense University, Taiwan (e-mail: superalf@gmail.com).

TABLE I: Service Models of Cloud Computing [1]

Service Name	service content
Infrastructure as a Service（IaaS）	IaaS is integrated as an infrastructure available for hire by individuals, companies, institutions, etc., to reduce the cost of the acquisition and management of access
Platform as a service（PaaS）	PaaS provides cloud application development environment, but does not control the operating system, hardware or network infrastructure operations.
Software as a service（SaaS）	Software developers and companies can develop freely, creatively and face users worldwide to provide software services

Server virtualization is a key technology in IaaS service model and is usually used as an idle computer resource available in physical servers, and number of virtual machines (VMs) is running on it without the users hardly experiencing performance reduction. Recently, its services have been expanded to improve the performance issues with distributed platforms. Hadoop is introduced as one of the most important cloud computing frameworks.

TABLE II: Survey of HDFS Small File Problem Solution

Name	Contribution	Level	File Type
HAR(2009) [3]	file merge	Hadoop	Small
Liu X (2009) [4]	The use of regional relevance merge files	Hadoop	Small
SequenceFile (2011) [5]	Serialization compression	Hadoop	Small
MapFile (2011) [6]	Index serialization compression	Hadoop	Small
Zhang(2012) [7]	Virtual machine load balancing	VM	Large / Small

Job scheduling and configuration strategy of Hadoop would significantly affect its performance; Data size and block number of files split depend on Hadoop infrastructure would affect its I/O speed. Number of virtual machines deployment and distributed resource allocation and adjustment would affect Hadoop throughput; Network route and bandwidth of Interconnection among virtual machines would affect its transmission efficiency. However, evaluating

DOI: 10.7763/IJIEE.2014.V4.487

performance of Hadoop cloud computing platform is complicated and not well documented. In this paper, the research is focused on the performance factors and model of HDFS and MapReduce on Hadoop. The following of this paper is organized: Section II is an overview of the Hadoop distribution file system (HDFS) and a brief introduction to the MapReduce programming model. The proposed Comprehensive Performance Rating (CPR) scheme is described in Section III. Section IV is the discussion on the experimental setup and results. In Section V, future research directions are concluded.

II. Related Work

A. Hadoop Distributed File System (HDFS)

Hadoop is a project develops open-source software for reliable, scalable, distributed computing. It consists four module: 1) MapReduce, a system for parallel processing of large data sets; 2) HDFS, a distributed file system that provides high throughput access to application data; 3) YARN, a framework for job scheduling and cluster resource management; 4) Common, the common utilities that support the other Hadoop modules.

This paper is using the CPR (Comprehensive Performance Rating) method to create an extension of YARN to configure MapReduce and HDFS. The Hadoop Distributed File System (HDFS), an Apache Hadoop subproject, is a highly fault-tolerant distributed file system designed to provide high throughput access to application data and is suitable for applications with extremely large data files (Petabytes and up) on a distributed network of computers. Architecture of HDFS [8] is shown as Fig. 1.

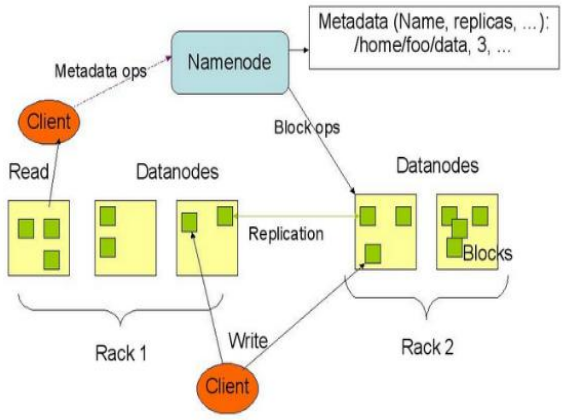


Fig. 1. HDFS architecture [8].

A HDFS cluster consists two kinds of nodes: (a) a single NameNode that works on master server and maintain the metadata structure of HDFS namespace includes opening, saving, retrieving, closing and renaming files and directories. It determines the mapping between files and blocks to DataNodes, and (b) a number of DataNodes manage storage attached to the nodes. The DataNodes serve read and write requests from the HDFS clients. Data file is split into one or more blocks, and these blocks are stored in a set of different DataNodes. The DataNodes also perform block creation, deletion, and replication upon instruction cooperating with the NameNode.

There are four operation of typical jobs involving HDFS: 1) client send a read comment to the NameNode, the NameNode then send back metadata to the client in order to confirm the process; 2) NameNode send metadata to the DataNodes; 3) DataNodes read and write the metadata ; 4) requested data is sent back from the DataNodes to the client via network. The critical factors in the process need to be probed and adjusted in order to improve the total performance in cloud computing.

B. MapReduce Framework

MapReduce was developed by Google as a new framework for processing large data sets. The original MapReduce software framework is not publicly available, but several open-source alternatives such as Hadoop MapReduce do exist and is available for the public to use. The Hadoop MapReduce is the parallel computation mechanism that is used to realize the transparency about the complexity of parallelization, data distribution, fault-tolerance, and load balancing to developers, which can be used to focus on developing the actual algorithm.

Fig. 2 is the MapReduce framework. Two major functions, Map and Reduce are specified in user program respectively. Although neither function is explicitly parallel, many instances of these functions are allocated and executed concurrently by the framework.

A master node allocates Map and Reduce tasks to worker nodes for execute client program. The input files, stored in HDFS, are fetched and split into data chunks. The following steps are then required for MapReduce framework to complete the requested jobs:, 1) data chunks are distributed to parallel Map tasks for processing, then key/value records are generated and stored in intermediate files. No communication is allowed due to parallel Map processing tasks, 2) the output intermediate file are copied to Reduce tasks nodes, 3) according to the key/value, Sort function is then executed, 4) the Reduce function is accomplished, multiple outputs are then finally created.

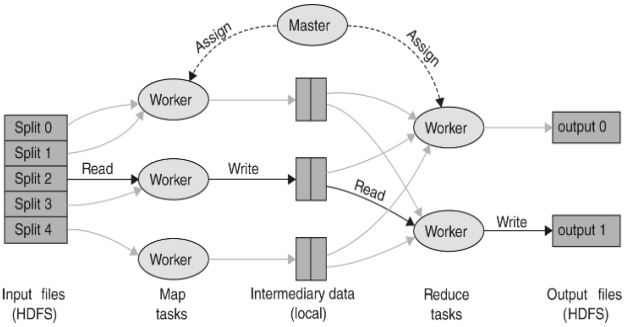


Fig. 2. MapReduce framework [9].

Data locality of Map/Copy/Sort/Reduce function is the one of critical factors in MapReduce processing, such as distance between storage and compute nodes that is across networks, racks or nodes. For Map executing phase, the node executing the Map tasks should be close to the node that stores the input data (preferably local to the same node). For Sort phase, it is required to move the intermediate data generated by the Map tasks to the Reduce tasks as their input. For Reduce phase, the node executing Reduce tasks should be close to the Map tasks nodes which generate the intermediate file used as Reduce tasks input. The data locality issues can cause a massive

network traffic, imposing a serious constraint on the efficiency of big data applications in cloud computing.

III. Comprehensive Performance Rating (CPR) Scheme

CPR system is described from three different views: architecture, metrics and model.

A. CPR Architecture

The proposed Comprehensive Performance Rating (CPR) system is designed to work in virtual clusters in MapReduce framework. Fig. 3 is shown to illustrate the architecture of cloud computing system. The architecture consists of two main modules: Performance Tuning Advisor (PTA) and Joint Performance Analyzer (JPA).

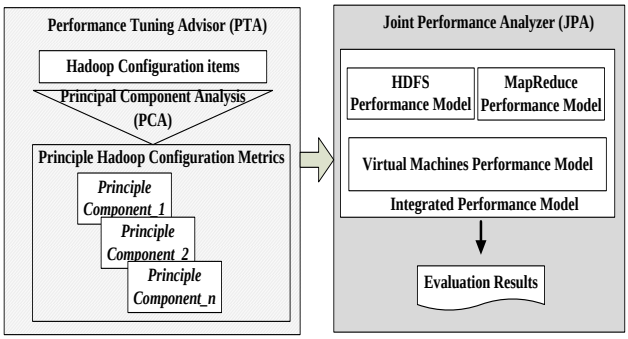


Fig. 3. System architecture of CPR scheme.

PTA module is used as a guidance for performance tuning to the integrated performance model in JPA. PTA is extended from [10] Principle Component Analysis (PCA) to fill out the critical Hadoop configuration metrics that strongly impact the workload performance from excessive configuration items. The JPA modules includes an integrated performance evaluated model composed of parameters and interference created by joining HDFS performance model and MapReduce performance model for virtual Hadoop clusters. The JPA would modify the values of CPR according the guideline from PTA and effectively produce the evaluation results.

$$M_{NN} = M_b \times N_{file} + (N_{file} + M_\beta) \times \sum_{i=1}^{N_{file}} \left\lceil \frac{L_i}{HBS} \right\rceil + M_\alpha \quad (1)$$

B. CPR Metrics and Hadoop Configuration Items

There are three categories of metric in CPR: The performance of virtual machine, HDFS and MapReduce metrics. The first category, critical virtual machine performance metrics are listed in Table III, which include CPU, memory, I/O, response time, and network.

TABLE III: VM Performance Model Parameters

Symbol	Expression
$T_{response}$	response time
$IO_{throughputs}$	I/O throughputs
r_{cpu}	CPU ratio
$tf_{network}$	Network traffic

The second category, HDFS performance metrics describe the access characteristics of distributed file system. The category includes NameNode, DataNode, file, memory, block, time, metadata, and network in HDFS. The symbols and expressions are as listed in Table IV.

TABLE IV: HDFS Performance Model Parameters [11]

Symbol	Expression
M_{NN}	File Number per KB of Memory of NameNode
N_{file},N_{map}	File number / Map number
M_α,M_β	Namenode / MapBlock Memory cost
HBS, L_i	Block size / File of lengths
M_b, M_r	Block/ Replicas Memory cost of each file
T_{HDFS}	HDFS time cost of each file
δ_{CN}	a client sends a read command to NameNode time cost of each file
δ_{NC}	NameNode looks up the metadata of the requested file in memory time cost
δ_{CD}	the metadata is returned to the client of each file
$\delta_{metadata}$	Metadata time cost of each file
δ_{disk}	Disk read / write time cost of each file
$f_{network}$	Networks traffic time cost of each file

The last category, MapReduce performance metrics are about transforming problems into parts of Map and Reduce, which includes job, time, simultaneously number, throughput, network bandwidth, I/O, buffer and rack. The symbols and expressions are listed in Table V.

C. CPR models

CPR model describes the behaviors and characteristics of CPR system, which includes NameNode model, HDFS model and MapReduce model .In NameNode model, the consumed memory of NameNode when accessing a file from HDFS as mentioned in [11] for ease of capturing the performance workload of HDFS is expressed as formula (1).

In formula (1), when client is sending a request of file to Hadoop system, an access command is sent to NameNode. The metadata of the requested file is retrieved from the memory and sent back to the client. Access command is sent from the client to a corresponding DataNode. The data blocks of the files are fetched by the DataNode, the blocks are then sent to the client afterward. The process time required for reading an access file from HDFS is derived as formula (2).

$$T_{HDFS} = N_{file}(\delta_{CN} + \delta_{NC} + \delta_{CD}) + \sum_{i=1}^{N_{file}} \delta_{metadata} + \sum_{i=1}^{N_m} \delta_{disk} + \sum_{i=1}^{N_m} f_{network} \left(\frac{L_i}{speed_i} \right) \quad (2)$$

The sub-model that represents the performance of a MapReduce Job includes the execution time from Map time (T_m), Copy time (T_c), Sort time (T_s) and Reduce time (T_r) described in [12] in formula (3) to (6) respectively.

$$T_m = \frac{D_{blk\_size}}{P_m} \times N_{m\_wvs} \quad (3)$$

$$T_c = \max\left(\frac{MapOutSizeForLocalReduce}{LocalRackCopySpeed}, \frac{MapOutSizeForRemoteReduce}{RemoteRackCopySpeed}\right) \quad (4)$$

$$T_s = \begin{cases} \frac{2 \cdot \lceil \frac{N_{reduce}}{N_{servers}} \rceil}{S_{disk_{write}}} \cdot \lambda_{N_{sortpaths}} \left(\frac{N_{in_{blks}} \cdot D_{blk_{size}} \cdot r_{map_{io}}}{N_{reduces} \cdot D_{sortbuf}}, D_{sortbuf} \right) \\ , \frac{N_{in\_blks} \cdot D_{blk\_size} \cdot r_{map\_io}}{N_{reduces} \cdot D_{sortbuf}} > 1 \\ 0 \qquad , otherwise \end{cases} \quad (5)$$

$$T_r = \max(T_{r\_proc}, T_{r\_rep}) \quad (6)$$

TABLE V: MAPREDUCE PERFORMANCE MODEL PARAMETERS

Symbol	Expression
T_j, T_m, T_c, T_s, T_r	Accumulated actual job/ map / copy/ sort / reduce wave execution time
T_{r_rep}	Total reduce output writing time
T_{r_proc}	Total reduce processing time
$N_{m_{slots}}, N_{r\_slots}$	Number of simultaneously executable maps/reduce in a server
$N_{m_{wvs}}, N_{servers}$, $N_{racks}, N_{reduces}$	Number of map/ servers/ reduce/rack in a Cluster.
D_{blk_size}	Data block size
N_{in_blks}	Number of map input data blocks
P_m, P_r	Average map/ reduce throughput per server core
$N_{cp\_threads}$	Number of copy i/o threads per reducer node
B_{local}, B_{remote}	Network Bandwidth of intra-switch connection
S_{cp_thread}	Theoretical maximum copy speed per copy thread
S_{r_rep}	Theoretical maximum output replication speed per reduce
S_{disk_write}	Sequential disk write speed
$r_{map_{io}}, r_{reduce\_io}$	Map/Reduce output to input size size ratio
$N_{replicas}$	Number of replicas in HDFS
$N_{rep_{locals}/N_{rep\_remotes}}$	Number of local/remote rack replicas
$D_{sortbuf}$	Sort buffer size for copy

IV. CPR SYSTEM EXPERIMENTAL AND COMPARISONS RESULTS

A. Experiment Setup

Virtual Hadoop cluster was set up in this research, consisting of twenty homogeneous nodes with hardware and software specified in Table VI. A virtual machine was setup to be a master node, served as both Namenode and JobTracker. The other 19 VMs were setup to be slave nodes, acted as both DataNode and TaskTracker, Considering the hardware resource, TestDFSIO and WordCount benchmark tool with different test data size and cluster size on Hadoop cluster were executed. The following experiments are carried out on each dataset:

TABLE VI: EXPERIMENT SETUP

Server	CPU	INTEL Xeon E3-1230V2 quad-core at 3.3GGHz
	Memory	32GB
	Disk	1000GB
Software	OS	Linux Ubuntu 12.04_32bit
	Hadoop	Hadoop 1.0.4
	Java	Sun JDK 6u21

1) Run TestDFSIO read test with data sizes of 1.5GB, 3GB, 6GB and 12GB
2) Run WordCount benchmark whose cluster sizes are 5 nodes, 10 nodes and 20 nodes;
3) Run WordCount benchmark whose numbers of Map task are 2, 4 and 6, respectively.

B. Results Discussion

The first experiment were designed to compare the execution time with different cluster size with 5,10 and 20 nodes using the TestDFSIO read test. The result is shown in Fig. 4 and it is worthwhile to note, due to the computing capacity bottleneck, the work time is almost identical in 20-nodes and 10-nodes cluster.

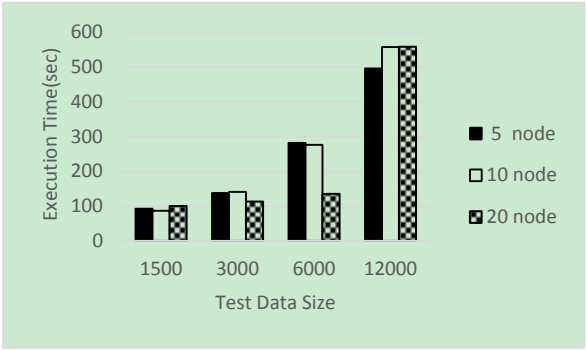


Fig. 4. TestDFSIO read test measured excution time for difference test data size and cluster size.

Due to parallel processing, the sum of map and reduce task was greater than the total job time. The greater the difference, the higher the parallelism and the higher the performance. In Fig. 5, the experiment results show that the ratio of Map task execution time to total job execution time was high under different cluster sizes, which means Map tasks occupy most of the total job execution time, therefore increasing the parallel processing in Map task is more efficient than increasing the parallel processing in Reduce task.

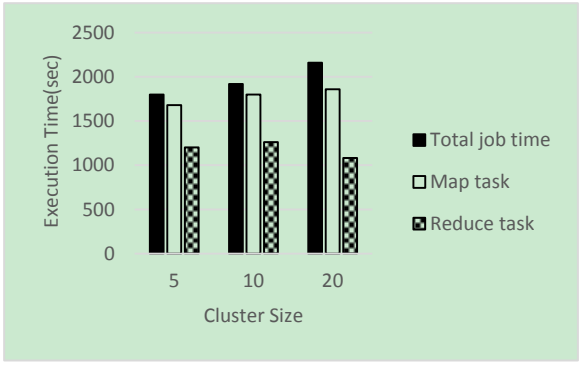


Fig. 5. The ratio of Map task execution time to total job execution time for difference cluster size.

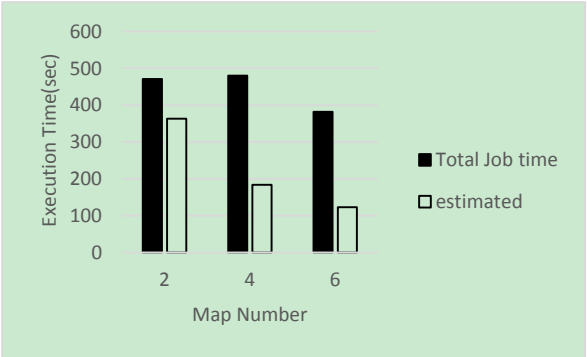


Fig. 6. WordCount benchmark estimated and measured total job time comprison for different map tasks.

The next experiment is designed to test the execution time within different task numbers, which determines the tasks that could be executed concurrently during the Map phase. By default, each Map task processes corresponding to one split input, therefore more parallel Map is brought from the increasing of independent input splits. Result is shown in Fig. 6, both the estimation and resulted execution time were deceased with the number of Map tasks increasing.

V. Conclusion and Future Work

and evaluating scheme to explore the problems, causes and solutions of VMs interference with data locality and excessive configuration parameters comprehensively. The proposed CPR scheme is based on the integrated performance model and the joint performance evaluator is guided by a performance tuning advisor to effectively process the performance tuning work.

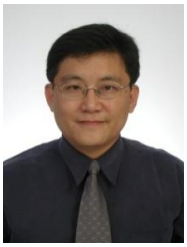
The experimental results demonstrated the performance of proposed scheme and compared the measured execution time with estimated execution time of proposed scheme. A possible extension of our work could be automatic adjustment of principle Hadoop configuration metrics for optimizing workloads composed of MapReduce job in the cloud computing.

Acknowledgment

This work was supported by the National Science Council of Republic of China under grant NSC 102-2221-E-606-008.

References

[1] NIST. [Online]. Available: http://www.nist.gov/
[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. the nineteenth ACM Symposium on Operating Systems Principles*, New York, NY, USA, 2003, pp. 29–43.
[3] Hadoop Archives. (2009). Hadoop archives guide. [Online]. Available: http://hadoop.apache.org/common/docs/current/hadoop_archives.html
[4] X. Liu, J. Han, Y. Zhong, C. Han, and X. He, "Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS," in *Proc. IEEE International Conference on Cluster Computing and Workshops, CLUSTER '09*, 2009, pp. 1–8.
[5] Sequence File. (2011). Sequence file. [Online]. Available: http://wiki.apache.org/hadoop/SequenceFile.
[6] MapFile. (2011). Mapfile api, [Online]. Available: http://hadoop.apache.org/common/docs/current/api/org/apache/hadoop/io/Map
[7] Z. Zhang, L. Xiao, Y. Li, and L. Ruan, "A VM-based resource management method using statistics," in *Proc. 2012 IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS)*, 2012, pp. 788–793.
[8] Hadoop Distributed File System (HDFS). [Online]. Available: http://hadoop.apache.org/hdfs/.
[9] G. Pratx and L. Xing, "Monte Carlo simulation of photon migration in a cloud computing environment with MapReduce," *Journal of Biomedical Optics*, vol. 16, no. 12, pp. 125003–1250039, 2011.
[10] H. Yang, Z. Luan, W. Li, and D. Qian, "MapReduce Workload Modeling with Statistical Approach," *Journal of Grid Computing*, vol. 10, no. 2, pp. 279–310, Jun. 2012.
[11] B. Dong, Q. Zheng, F. Tian, K.-M. Chao, R. Ma, and R. Anane, "An optimized approach for storing and accessing small files on cloud storage," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1847–1862, Nov. 2012.
[12] J. Han, M. Ishii, and H. Makino, "A Hadoop performance model for multi-rack clusters," in *Proc. 2013 5th International Conference on Computer Science and Information Technology (CSIT)*, 2013, pp. 265–274.



Fong-Hao Liu received M.S. degree in information engineering from National Chung Kung University (NCKU), Taiwan, in 1990 and PH.D degree in electronic engineering from National Taiwan University of Science and Technology (NTUST), Taiwan in 2000. He joined the department members of Information Management Department of National Defense University as associate professor in 1990. His research interests are software engineering, computer networks and network security.

Ya-Ruei Liou received his B.S. degree in Computer Science from Shih Hsin University, Taiwan, in 2012. He is currently working toward the M.S. degree at Management College of National Defense University, Taiwan. His research interests include virtualization technology, hadoop performance control and cloud computing.

Hsiang-Fu Lo received B.S. and M.S. degree in computer science from Management College of National Defense University, Taiwan, in 2000 and 2004. He is currently working toward the Ph.D degree at Tamkang University, Taiwan. His research interests include wireless video transmission, cloud computing and computer networks.

Ko-Chin Chang received the M.S. degrees from Department of Electrical and Electronic Engineering, Chung Cheng Institute of Technology, National Defense University, Taiwan, in 2003, and is currently pursuing the Ph.D. degree in the Department of Electrical and Electronic Engineering, National Defense University, Taiwan. His research interests include image and signal processing, data hiding, and image compression. His recent research has focused on high capacity image steganography.

Wei-Tsong Lee received B.S., M.S. and Ph.D degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan. In 2003, he joined the department members of Electrical Engineering of Tamkang University as an associate professor, and reached professor in 2007. From 2010, he is the chairman of Electrical Engineering Department. His research interests are embedded system, computer architecture, micro-processor interface and computer networks.