MapReduce Based Experimental Frame for Parallel and Distributed Simulation Using Hadoop Platform

Byeong Soo Kim, Sun Ju Lee, and Tag Gon Kim
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology
Daejeon, 305-701, Republic of Korea
E-mail: {bskim, sjlee}@smslab.kaist.ac.kr
tkim@ee.kaist.ac.kr

Hae Sang Song
Department of Computer Engineering
Seowon University
Cheongju, 361-742, Republic of Korea
E-mail: hssong@seowon.ac.kr

KEYWORDS
MapReduce, experimental frame, parallel and distributed simulation, Hadoop, system analysis.

ABSTRACT

Simulation-based experiment of complex systems is a time consuming-job. Parallel and distributed simulation is one of the methods to reduce the simulation time. To simulate and analyze the system with this method, it is required to design a suitable experimental frame. Therefore, this paper proposes a MapReduce based experimental frame for the parallel and distributed simulation. Because Hadoop MapReduce is the most widely used parallel and distributed computing platform, we use it to design the experimental frame. In our work, the 'map' of MapReduce automatically generates and simulates the system, and the 'reduce' of MapReduce collects and analyzes the result. We can reuse the existing large scale Hadoop clusters without any modification of the platform, so it is easy to set-up and use the experimental frame. This paper presents an air defense simulation to show the usage and speed up with a 16-node Hadoop cluster.

INTRODUCTION

To analyze characteristics of a system, relationships must be drawn between input parameters and performance indices of the system. The more complex the system is, the more researchers need time and effort to draw how the inputs affect the performance indices. For example, assume that there are various parameters like number of missiles, speed of missile, accuracy of missile, range of radar, attack power, and decision making time for the simulation of an air defense system (Cho et al. 2007). If each parameter has five levels, the system could have over 10,000 scenarios with full factorial design (Antony 2003). Then, there needs to be over 10,000 minutes to simulate all the scenarios of the system, even if each simulation takes only one minute. It is also a time-consuming job to collect and analyze the results after the simulation.

Therefore, many researches have attempted faster simulation methods to remedy this problem. Generally, parallel and distributed simulation approaches have been widely used to reduce time-consuming phenomenon (McGeoch 1992). To simulate the system with parallel and distributed environment, it is required to design an appropriate experimental frame. An experimental frame is a specification of the conditions under which the system is experimented with (Zeigler et al. 2000). It is composed of a generator, which generates the input segments and a transducer, which collects and analyzes the output segments of the system.

In simulation fields, there is research adapting parallel and distributed simulation techniques for faster data collection of the simulation: DEXSim (Choi et al. 2014), CR-PADS (Bononi et al. 2005), and mJADES (Rak et al. 2009). They provide efficient simulation environments with best use of distributed hardware resources, however they did not consider faster data analysis. In other words, they did not provide an experimental frame for faster data collection and faster data analysis. Furthermore, since the previous studies were developed in their specific environment, they spend much time and cost to expand the distributed machines.

In our approach, parallel and distributed simulation is used to design and simulate a system; it is also used to gather and analyze the results after the simulation. We use the Hadoop platform for implementation of the proposed work for the convenience of environment construction. Hadoop is the most widely used platform for distributed computing (White 2012). It is a scalable, common, and reliable platform compared to other platforms used for the previous studies. Furthermore, MapReduce, a distributed computing framework of Hadoop, is appropriate to adapt the concept of generator and transducer. Although there is research about simulation using MapReduce (Decraene et al. 2011; Jakovits et al. 2011; Pratx and Xing 2011), there is no research about experimental frames for the simulation of legacy simulators.

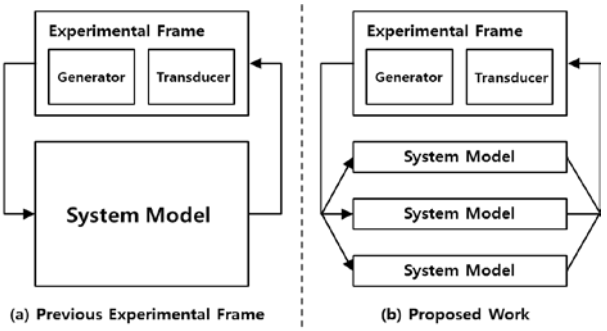

Figure 1: Previous and Proposed Experimental Frame

Proceedings 28th European Conference on Modelling and Simulation ©ECMS Flaminio Squazzoni, Fabio Baronio, Claudia Archetti, Marco Castellani (Editors)
ISBN: 978-0-9564944-8-1 / ISBN: 978-0-9564944-9-8 (CD)

In this paper, we propose a new experimental frame for parallel and distributed simulation using the Hadoop platform (Figure 1-(b)). The proposed work provides an experimental frame for efficient experimental design and result analysis. It can also be used for simulation-based optimization (Hong et al. 2013). Because it reuses the existing large-scale Hadoop clusters without any modification to the platform, users do not need to set the distributed environment. Basically, Hadoop provides reliability and powerful load balancing; users have only to take advantage of Hadoop platform.

This paper is organized as follows. Background materials about the Hadoop platform and Hadoop Streaming are briefly introduced. Then our proposed experimental frame using the Hadoop platform and simulation process are described. Finally, a case study using an air defense simulator is provided.

HADOOP

Hadoop is a representative big data platform developed by the Apache Software Foundation. It is an open source implementation for reliable, scalable, large scale distributed computing (White 2012). Hadoop consists of MapReduce and Hadoop Distributed File System (HDFS). MapReduce is a distributed computing framework for large scale data processing. HDFS is a distributed file system that stores data reliably using commodity machines (Shvachko et al. 2010). The Hadoop platform is fault-tolerant for hardware and network failures, and provides efficient resource management.

MapReduce

MapReduce is a framework for large-scale distributed data processing based on the divide and conquer paradigm. MapReduce works by breaking the processing into map and reduce (Dean and Ghemawat 2008). Map and reduce are executed in parallel on the different machines within the Hadoop cluster by MapReduce framework. Map performs filtering and sorting operations, and reduce performs summary operations. The user can specify map/reduce functions, and types of input/output.

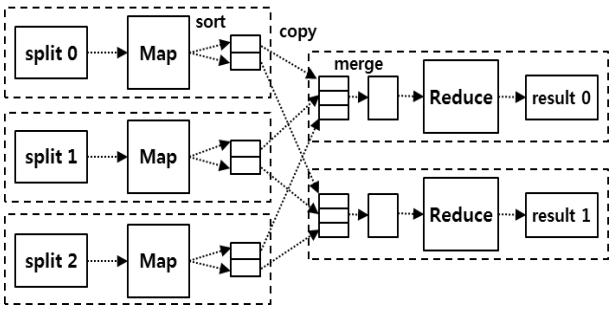

Figure 2: Overall Process of MapReduce

Figure 2 illustrates the overall process of MapReduce. Input data stored on the HDFS are split into fixed-size blocks, and each block is allocated to a map. Then user-specified map processes each key-value pair in the block; and outputs the result as a list of key-value pairs. The output of the map is partitioned by the key, and the grouped records are transferred to the different reduces, respectively (called shuffle). Then, the transferred records are merged and sorted in the node which a reduce task located. Each reduce sequentially reads key-value pairs, and processes them by the user-specified reduce function. Finally, the output records of the reduce are written to the HDFS.

$$Map\ (k_1, v_1) \rightarrow list\ (k_2, v_2)$$
$$Reduce\ (k_2, list\ (v_2)) \rightarrow list\ (v_3)$$

Hadoop Streaming

Because MapReduce applications are executed in the form of source code, it is difficult to run an executable legacy simulator on the MapReduce framework. Therefore, an interface is required to run one on the MapReduce framework. Developers can implement the interface directly, but for convenience, the Hadoop platform provides a utility called Hadoop Streaming.

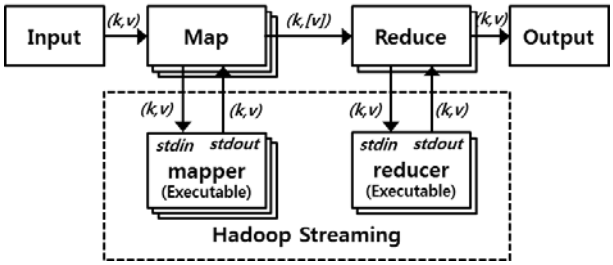

Figure 3: Overview of Hadoop Streaming

Hadoop Streaming is a Hadoop utility (application) which allows creating and running MapReduce jobs with any executable program as the Mapper and Reducer. Executable programs communicate with Hadoop Streaming through Unix streams (Figure 3). They read the input key-value pairs from standard input (stdin) line by line, and emit the output key-value pairs to standard output (stdout). In this paper, we use Hadoop Streaming to implement the proposed work.

MAPREDUCE BASED EXPERIMENTAL FRAME

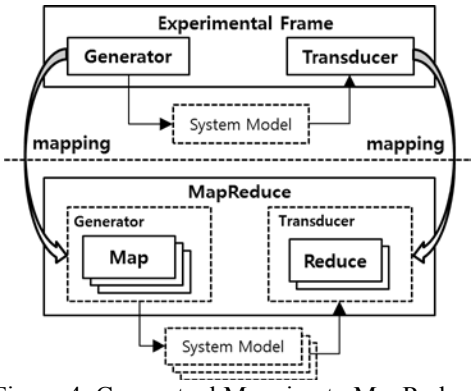

Figure 4: Conceptual Mapping to MapReduce

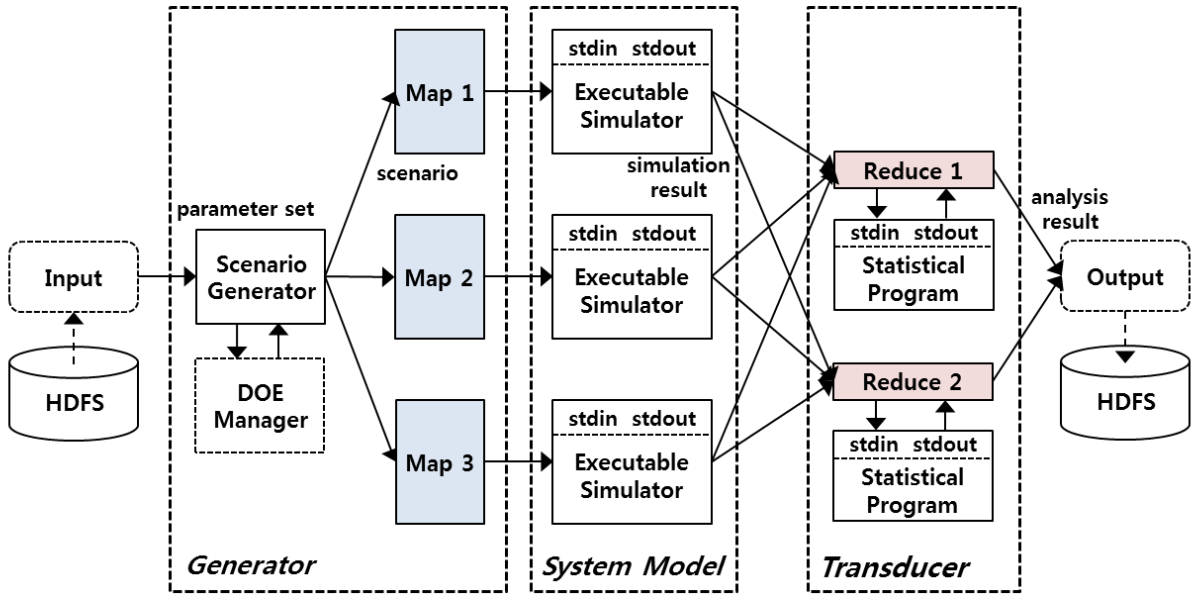

Figure 5: Overall Structure of Experimental Frame

In this section, we propose an implementation of an experimental frame adapting Hadoop MapReduce. Figure 4 shows the conceptual mapping between the experimental frame and the MapReduce framework. In our approach, map performs the role of generator which generates scenarios and allocates them to the system model. The reducer performs the role of transducer which collects the output of simulations and analyzes the results.

Overall Structure

The generator of the proposed experimental frame is composed of scenario generator, design of experiment (DOE) manager, and map of MapReduce framework (Figure 5). The scenario generator makes scenarios from input parameter set using DOE manager. The DOE manager handles the design of experiment, but in this paper, only full-factorial design (Antony 2003) can be used. Later, it is possible to apply several design of experiment methods extending the DOE manager. The outputs of the scenario generator (all scenarios) are automatically split into an individual scenario by MapReduce framework, then each scenario is allocated to each map. MapReduce framework assigns a map to each CPU core of individual machine in Hadoop, and provides efficient load balancing and resource management. Consequently, efficient and faster data collection are possible with parallel and distributed environment of Hadoop.

The transducer is composed of statistical program and reduce of MapReduce framework. The statistical program, which can be a commercial (e.g., R project for statistical computing) or a user-developed application, processes and analyzes the simulation output. The results of the analysis can be statistical values of the simulation, optimized input parameters or extracted internal state. The reduce collects and saves the output to the HDFS. It is executed in parallel like the map, therefore faster data analysis is possible using the experimental frame. In this proposed work, the legacy simulator has to receive an input scenario from stdin, and emit the output to stdout in the form of key-value pair. It is the same with the statistical program.

Simulation Process

The simulation process is composed of two phases: the preparatory phase, and the main phase (Figure 6). The preparatory phase generates scenarios and sets up the experimental environment. The main phase executes the simulator and analyzes the simulation result. The detail processes are as follows.

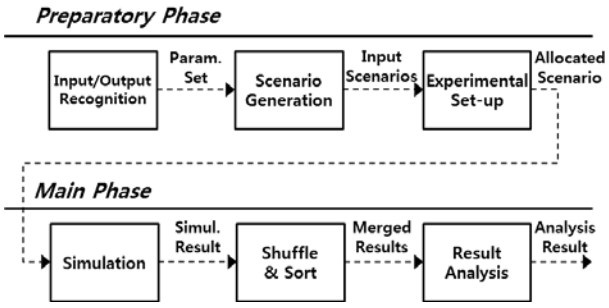

Figure 6: Simulation Process: Preparatory Phase and Main Phase

Preparatory Phase

In the preparatory phase, we specify object/attribute and performance index following the objective of the simulation. We use the Object-Performance Index (OPI) matrix to recognize the relation between the object and the performance index (Kim and Sung 2007) and make the Parameter Set shown in Table 1. The experimental frame automatically generates scenarios adapting the design of the experiment and writes the scenario file to

the HDFS. It also saves the simulation model and simulation engine (in this paper, DEVS simulation model and DEVSimLinux binary files) to the local file system (LFS) of each node. Then, the experimental frame sets up the experimental environment: It sets the total number of map tasks following the number of scenarios, the number of reducers, and the number of task slots. Finally, the MapReduce framework splits the scenario file into the number of maps and allocates each scenario to each map task.

Table 1: Specification of Input / Output

Type	Example
Parameter Set	(Attribute 1 : range of value, Attribute 2 : range of value, … Performance Index : threshold value)
Input Scenario	(Scenario 1, Parameter 1 : value, Parameter 2 : value, …)
Simulation Result	(Scenario 1, Result 1 : value, Result 2 : value, …)
Analysis Result	(Scenario 1, Parameter 1: value, … Performance Index : value)

Main Phase

In the main phase, each map executes the simulator stored in the LFS with the allocated scenario. The simulator reads the scenario in the form of the key-value pair. After the simulation is finished, it emits the result of the simulation in the form of the key-value pair to its own map. Then the simulation results of all the maps are sent to the reduce through the shuffle process of MapReduce. The reduce merges and sorts the results from the maps, and the developed statistical program analyzes the results. Finally, they are merged and written in the HDFS by the MapReduce framework. The main phase is automatically performed by the MapReduce framework, so the user does not need to manage the simulation after the execution of the experimental frame.

CASE STUDY: AIR DEFENSE SIMULATOR

This section presents a case study to show the efficiency of the proposed experimental frame. The experiment was conducted on a homogenous Hadoop cluster of 16 machines, which consisted of one master machine and 15 slave machines. Each machine had quad-core Intel i5-3550 CPUs running at 3.3 GHz, Samsung DDR3 4 GB RAM, and Samsung HDD 500 GB, running on Ubuntu-12.04 32bit. We used Hadoop-1.1.2. The machines were connected to a gigabit Ethernet switch with two trunked gigabit ports per machine.

Target Simulator

The target simulator is an air defense system simulator which detects and nullifies missiles from an enemy. It is modeled using DEVS formalism, and is running on the DEVSimLinux, discrete event system simulation engine (Kim et al. 2011). The simulator is composed of four parts; missile, radar, weapon, and C2A (Command & Control and Alert) system model. The C2A system receives target information from radars, makes decisions based on algorithms, and sends attack orders to the air defense weapon systems.

The simulator represents the situation of defending a base against missiles and analyzes the effectiveness of the air defense system for various parameters. When missiles are fired, installed radars detect the missiles and send the information to the C2A system. Then the C2A system assigns weapon systems according to the algorithms and orders an attack to defend the base. Finally the simulator measures the defense rate in accordance with the success or failure of the attack.

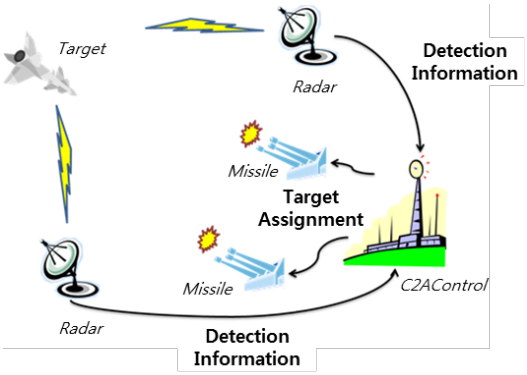

Figure 7: Air Defense Simulator

Experimental Design

We designed experiments to acquire the scenarios whose defense success rate was more than 80% for the various parameters. To find the desired scenarios, full-factorial design is needed for the input parameters and its values. So, the generator was implemented to do full-factorial design for the input. Each scenario is executed 30 times to calculate its defense success rate. Table 2 shows six parameters and four values per parameter, as an input for the generator. Therefore, the generator makes a total of 4,096 scenarios and 122,880 simulation runs are required.

Next, we designed a transducer to find the desired scenarios among all scenarios. The transducer can utilize statistical programs, but in this case study, we implemented just a simple filter to find the desired scenarios.

Table 2: Parameter Set of Air Defense Simulator

Parameter	Value	Level
Radar Detection Range (km)	3 ~ 6	4
Number of Radars	1 ~ 4	4
Period of C2A (sec)	1.0 ~ 2.5	4
Weapon Range (km)	1.5 ~ 3.0	4
Weapon Accuracy Rate (%)	60 ~ 90	4
Number of Weapons	3 ~ 6	4
Total Scenarios	4,096 ($=4^6$)	

Experimental Result

File: /output/streaming/part-00000

Goto : /output/streaming go

Go back to dir listing
Advanced view/download options

```
No scenario is assigned to this task!!
Scenario1 : OXXOOXOOOXOOOXOOXOOOOOXOXOOOOO (AverageTime: 85.34, SurvivalRatio: 0.73333
Scenario2 : XXOOXXXOOOXOXOOOXOOOOOXOOXXOXO (AverageTime: 85.4733, SurvivalRatio: 0.6)
Scenario3 : OOXOOOOOOOOOOOOXXXOOOOXOOOOOO (AverageTime: 85.2867, SurvivalRatio: 0.833
Scenario4 : OXOXOOOOOOOOOOXXXXOOOXOOOXOXXO (AverageTime: 85.4533, SurvivalRatio: 0.666
Scenario5 : OOOOOOOOOXOOOXOOOOOOOOXOXOXOO (AverageTime: 85.28, SurvivalRatio: 0.83333
Scenario6 : OOOOOOOOOOXOXXOOOOXXOXOOOOOOO (AverageTime: 85.3267, SurvivalRatio: 0.8)
Scenario7 : OOOOOOXOXOOOOOOOOOOOOOXOXXOOO (AverageTime: 85.3067, SurvivalRatio: 0.833
Scenario8 : OOOOXOOXOOXOXXOOXOOOXXXOOXXXOO (AverageTime: 85.4, SurvivalRatio: 0.6)
```

Figure 8: Experimental Result of Air Defense Simulator Stored in HDFS

After the experiment completed, we got the result stored in HDFS as shown in Figure 8. Since Hadoop provides a monitoring tool for MapReduce and HDFS, it is convenient to check the progress of the experiment using this tool. The center of Figure 8 shows the experimental results: scenario number, performance index and statistical values such as simulation time. We can easily find the desired scenarios from the large number of scenarios.

Table 3: Execution Time of Total Experiment

	Hadoop	Single Node
Number of Scenarios	4096	4096
Execution Time (sec)	753	5 × 4096 = 20480
Execution Time with 30 Cores (sec)	753	20480 ÷ 30 = 680.67
Speed up (times)	20480 ÷ 753 = 27.20	
Utilization Rate	680.67 ÷ 753 = 0.90	

Also, we can analyze how much the proposed experimental frame can reduce the execution time. We compared total execution time of the Hadoop platform with single node. In this experiment, two simulators can be executed simultaneously in one node because the number of map slots is 2. So, theoretical speed up of the proposed work should be 30 times with 15 slave nodes. However, the maximum speed up was only about 27 times as shown in Table 3 due to overhead of the Hadoop platform. Also, Figure 9 shows that the more the scenarios increase, the more the utilization of the experimental frame also increases. This is because the ratio of Hadoop overhead becomes smaller, as the number of scenarios increases. Consequently, we know that the proposed experimental frame is more efficient for the simulation with larger numbers of scenarios.

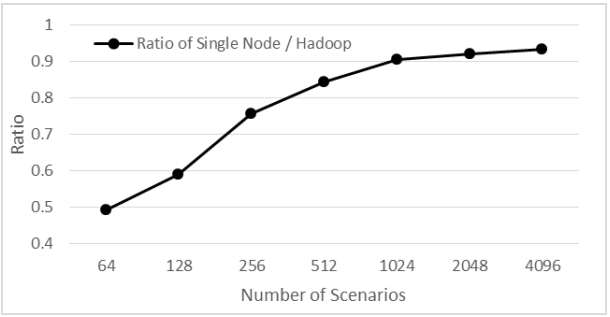

Figure 9: Utilization Rate according to Number of Scenarios

CONCLUSIONS

This paper proposes a MapReduce based experimental frame for parallel and distributed simulation using the Hadoop platform. Because simulation with a large number of scenarios requires much time to execute and analyze, the proposed work provides a generator for scenario generation and distributed execution of the simulator, and a transducer for distributed result analysis. In the proposed experimental frame, each generator/transducer is assigned to a CPU core by the MapReduce framework; faster data collection and data analysis are possible. The proposed work is also compatible with any modification or set-up of existing Hadoop cluster.

In this paper, we apply an air defense simulator which was developed in DEVS formalism to show the usefulness of the proposed work. A total of 4,096 scenarios were simulated on the 16-node Hadoop cluster with 30 map slots. The proposed experimental frame is only 27-times faster than a single node due to overhead of the Hadoop platform.

For further work, we will implement the DOE manager supporting several designs of experiment methods, and the statistical program analyzing these design of experiment methods. Also, we will research optimization methods using our proposed work.

ACKNOWLEDGEMENT

This work was partially supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract. (UD110006MD)

REFERENCES

Antony, J. 2003. *Design of Experiments for Engineers and Scientists*, Butterworth-Heinemann.

B.G. Cho; D.Y. Kim; S.H. Kim; C. Youn. 2007, "Real-Time Distributed Simulation Environment for Air Defense System Using A Software Framework". *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, Vol. 4 No. 2 (April), 64-79.

Bononi, L.; M. Bracuto; G. D'Angelo; and L. Donatiello. 2005. "Concurrent replication of parallel and distributed simulations". In *Proceedings of Workshop on Principles*

of Advanced and Distributed simulation (Jun.1-3), 234-243.

Choi, C.B.; K.M. Seo; and T.G. Kim. 2014, "DEXSim: an experimental environment for distributed execution of replicated simulators using a concept of single simulation multiple scenarios." *SIMULATION: Transaction of The Society for Modeling and Simulation International*, Accepted to be published, Jan.

Dean, J., and S. Ghemawat. 2008. "MapReduce: Simplified Data Processing on Large Clusters." *Communications of the ACM*, Vol. 51 Issue 1 (Jan), 107-113.

Decraene, J.; F. Zeng; M.Y.H. Low; W. Cai; Y.Y. Cheng; and C.S. Choo. 2011. "Evolutionary Design of Experiments using the MapReduce Framework". *SCSC '11 Proceeding of the 2011 Summer Computer Simulation Conference*, 76-83.

Jakovits, P.; I. Kromonov; S.N. Srirama. 2011. "Monte Carlo linear system solver using MapReduce". *2011 Fourth IEEE International Conference on Utiliy and Cloud Computing* (Victoria, NSW, Dec.5-9), 293-299.

Jeonghee Hong; Kyung-Min Seo; and Tag Gon Kim. 2013. "Simulation-based optimization for design parameter exploration in hybrid system: a defense system example," *SIMULATION: Transactions of The Society for Modeling and Simulation International*, Vol. 89, No. 3 (Jan), 362-380.

Kim, T.G. and C.H. Sung. 2007 "Objective-driven DEVS Modeling Using OPI Matrix for performance Evaluation of Discrete Event Systems". In *Proceedings of the 2007 Summer Computer Simulation* (San Diego, USA, Aug.), 305-311.

Pratx, G. and L. Xing. 2011 "Monte Carlo simulation of photon migration in a cloud computing environment with MapReduce." *Journal of Biomedical Optics*, Vol. 16(12) (Dec).

Rak, M.; A. Cuomo; and U. Villano. 2012. "mJADES: Concurrent Simulation in the Cloud". In *Proceedings of 2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems* (Palermo, Italy, Jul.4-6), 853-860.

Shvachko, K.; H. Kuang; S. Radia; and R. Chansler. 2010. "The Hadoop Distributed File System". *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies* (Incline Village, NV, May 3-7), 1-10.

Tag Gon Kim; Chang Ho Sung; Su-Youn Hong; Jeong Hee Hong; Chang Beom Choi; Jeong Hoon Kim; Kyung Min Seo; and Jang Won Bae. 2011 "DEVSim++ Toolset for Defense Modeling and Simulation and Interoperation," *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology,* Vol. 8, No. 3 (July), 129 - 142.

White, T. 2012. *Hadoop – The Definitive Guide*, O' REILLY®, YAHOO!® PRESS.

Zeigler, B.P.; H. Praehofer; and T.G. Kim. 2000. *Theory of Modeling and Simulation*, Second Edition, Academic Press.

AUTHOR BIOGRAPHIES

BYEONG SOO KIM is a Ph D. candidate at the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST). His research interests include methodology for M&S of discrete event systems (DEVS), distributed simulation, and system design.

SUN JU LEE is a master's degree candidate at the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST). His research interests include methodology for M&S of discrete event systems (DEVS) and distributed file system.

TAG GON KIM is a professor at the Department of Electrical Engineering Korea Advanced Institute of Science and Technology (KAIST). He was the Editor-In-Chief for Simulation: Transactions for Society for Computer Modeling and Simulation International (SCS). He is a co-author of the text book, Theory of Modeling and Simulation, Academic Press, 2000. He published about 200 papers in M&S theory and practice in international journals and conference proceedings. He is very active in defense modeling and simulation in Korea.

HAE SANG SONG studied his MS.D and Ph.D courses in Electrical Engineering in Korea Advanced Institute of Science and Technology (KAIST). He worked for a couple of years in an R&D lab, IAE (Instituted of Advanced Engineering) in 1999-2000. He also worked in a venture company for about two years, and has been a professor of Dept. Computer Engineering of Seowon University, Korea, since 2002. His major interest resides in modeling simulation, analysis, and control of discrete event dynamic systems.