International Journal of Grid and Distributed Computing
Vol. 5, No. 3, September, 2012

An Empirical Study for Handling Scientific Datasets

Yunhee Kang and Heeyoul Choi

Baekseok University, Samsung Advanced Institute of Technology
yunh.kang@gmail.com, heeyoul@gmail.com

Abstract

Since the volume of data generated by a scientific data experiment has grown exponentially, new scientific methods to analyze and organize the data are required. Hence, these methods need to be used effective infrastructure composed of computing resources that are used for pre-processing and post-processing data. The demanding requirement has led to development of methods to reduce the size of dataset and to apply a new programming model and its implementation like MapReduce. In this paper, we describe an empirical study for handling the dataset of a scientific data experiment to support data transformation, which is an essential phase to handling large-scale data in scientific data experiments. In this experiment we show a way to optimize the dataset written in netCDF by a data reduction as a sub-setting method and to process the dataset about tornado outbreak in the US by Hadoop, a MapReduce framework. These methods can be applied to pre-processing and post-processing in scientific data experiments.

Keywords: *MapReduce, Scientific Data Experiment, Sub-Setting, Data Transformation*

1. Introduction

Many scientific applications require processes for handling data that no longer fit on a single cost-effective computer. Besides, scientific data experiments such as simulations are creating vast data stores that require new scientific methods to analyze and organize the data when preparing the data for a simulation and handling the result of the simulation. For such data experiments to handle scientific problems, analysis of large amounts of data is necessary. Hence, to tackle these challenges, we need to support effective infrastructure composed of computing resources that are used for pre-processing and post-processing data as well as analyzing data [8].

Pre-processing and post-processing in a scientific data experiment are tedious and error-prone works. In these processing jobs, transforming such content into a structured format for later analysis is a major challenge. Those scientific applications devote most of their processing time to I/O and movement of data. Parallel/distributed processing of data-intensive applications typically involves partitioning or subdividing the data into multiple segments which can be processed independently using the same executable application program in parallel on an appropriate computing platform, then reassembling the results to produce the completed output data [2, 8].

One approach for tacking large-scale data intensive analysis is to use a MapReduce framework. A MapReduce programming helps focus on the problem that needs to be solved since only the map and reduce functions need to be implemented, and its framework takes care of the burden of tedious and error-prone works, otherwise programmers would have to deal with lower-level mechanisms to control the data flow [3, 4, 5, 6].

In this paper, we describe an empirical study for handling a dataset of a scientific data experiment to support data transformation, which is an essential phase to handling large-scale data in scientific data experiments. In this experiment we show a way to optimize the dataset written in netCDF by data reduction as a sub-setting method and to process the dataset about tornado outbreak in the US by Hadoop [13], a MapReduce framework. These methods can be applied to pre-processing and post-processing in scientific data experiments.

The rest of paper is organized as follows: Section 2 describes the related works about a scientific data project named PolarGrid and a brief overview of MapReduce and Hadoop. Section 3 describes an overall architecture of the scientific data framework proposed. The case studies are described in Section 4. Conclusions follow in Section 5.

2. Related Works

2.1. PolarGrid: Scientific Data Project

The PolarGrid project [7] is an NSF-funded project that provides computing support for the Center for the Remote Sensing of Ice Sheets (CReSIS). CReSIS is primarily concerned with using Synthetic Aperture Radar (SAR) techniques to obtain information on the depth of the Greenland and Antarctic ice sheets and their underlying rock beds. The Polar Grid cyberinfrastructure consists of ruggedized laptops and clusters deployed in the field in the polar regions, and two large-scale clusters for detailed data analysis in the US. Figure 1 shows a trend of the growth of data volume in the PolarGrid. The trend is caused by widely providing of sensors. PolarGrid provides both in-the-field computing clusters for initial image processing (useful for finding problems with radar equipment, for example) and larger clusters at Indiana University for full-scale image processing needed to make community data products.

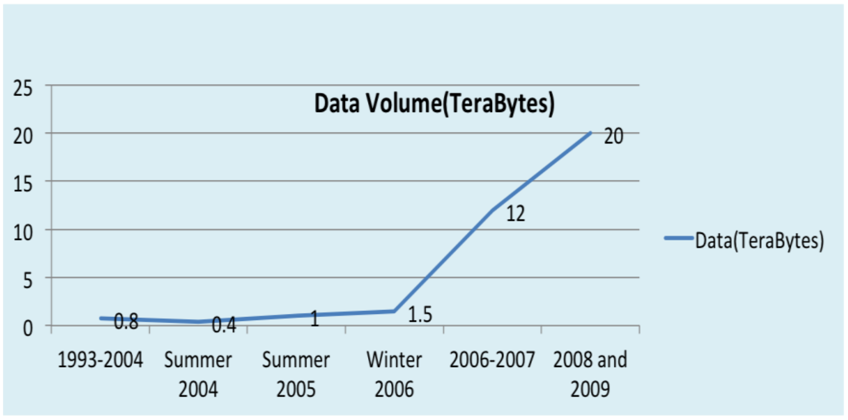

Figure 1. The Growth of Data Volume Gathered from the PolarGrid Project

2.2 MapReduce

MapReduce is an emerging programming model for a data-intensive application proposed by Google. MapReduce borrows ideas from functional programming, where a programmer defines map and reduce tasks to process a large set of distributed data. Traditional parallel applications are based on a runtime library that has some features of communication and synchronization [12]. The features provided by a runtime library are low-level primitive ones.

However, a MapReduce programmer is able to focus on the problem that needs to be solved, since only the map and reduce functions need to be implemented. Then, the framework takes care of the burden that the programmer would have to deal with lower-level mechanisms to control the data flow [2, 4]. Other key strengths of the MapReduce programming model are the high degree of parallelism combined with the simplicity of the programming model, and its applicability to a large variety of application domains [3,5]. This requires dividing the workload across a large number of machines. Note that the degree of parallelism depends on the input data size.

In this programming model, the computation takes a set of (key, value) pairs, and produces a set of output (key, value) pairs. It consists of two functions: Map and Reduce. These two functions have the following signatures:

$$Map :: (key1, value1) \rightarrow list((key2, value2))$$
$$Reduce :: (key2, list(value2)) \rightarrow list((key3, value3))$$

The map function processes the input pairs (key1, value1) returning some other intermediary pairs (key2, value2). Then, the intermediary pairs are grouped together according to their key values. After that, each group will be processed by the reduce function which will output some new pairs of the form (key3, value3).

We explain the map and reduce functions with the data transformation task with tornado outbreaks (See Section 4.2 for the details). Algorithm 1 gives a simplified representation of a map function used to process a set of records. A line represents a record of data associated with tornado outbreaks. The input records are mapped as key/value pairs with the key being the record id and the value being the series of records for each year. An intermediate reduce stage is used to construct the series of objects from the input key/value pairs. The input to the map function is a partition of the dataset represented by a multi-dimensional data to process. The map function extracts relevant data, which are associated with the specific variables in a record, by a filter function. In the map function, we emit the value that corresponds to a partial count over one instance.

```
Algorithm 1: Map()
<key> : line offset within the input file
<value> : line itself from the input file
// A line represents a record
// Extract year and a subset of variables in the record
 (year, occurrence) ← filter(line)
Emit(year, occurrence)
```

The MapReduce execution framework guarantees that all values associated with the same key are brought together in the reducer. In Hadoop, the reducer is presented with a key and an iterator over all values associated with the particular key. The values are arbitrarily ordered. In Algorithm 2, we simply need to sum up all counts associated with each year. The reduce function calculates the global sum, and then emits the final value with regard to a year.

```
Algorithm 2: Reduce()
<key> : year
<value> : iterator over the list of tornado occurrences
// Extract year and a subset of variables in the record
Emit(key, sum([list of occurrences])
```

Figure 2 shows the flow of data in the MapReduce processing. In a MapReduce application supported by a MapReduce library, all map operations can be executed independently. Each reduce operation may depend on the outputs generated by any number of map operations. All the reduce operations, however, can also be executed independently.

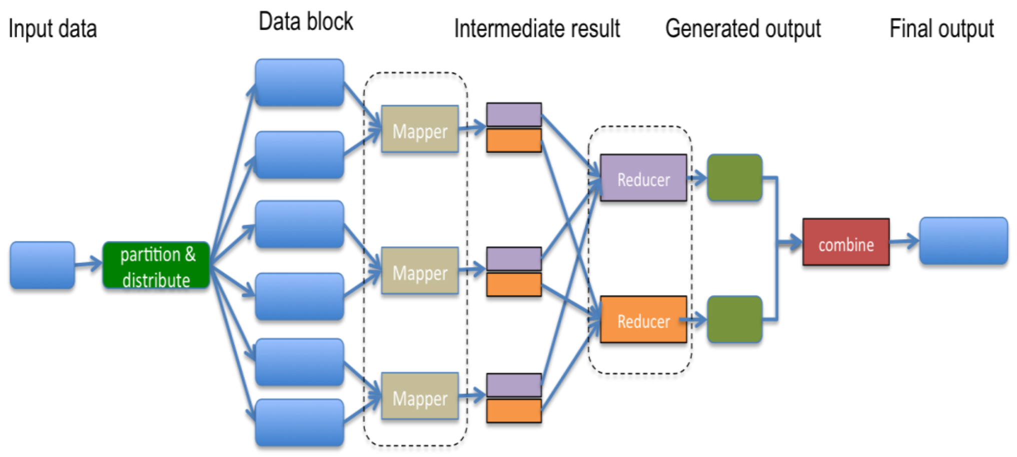

Figure 2. The Flow of Data in the MapReduce Processing

Hadoop [13] is an open source based on the MapReduce framework to run applications on large clusters built of commodity hardware from Apache. The Hadoop framework transparently provides applications with both reliability and data motion. Hadoop is an implementation of Map/Reduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster. In addition, it provides the Hadoop distributed file system (HDFS) that stores data on the computing nodes, providing a very high aggregate bandwidth across the cluster. HDFS is the primary storage system used by Hadoop applications. It creates multiple replicas of data blocks and distributes them on the computing nodes throughout the cluster to enable reliable, extremely rapid computations. Here are some properties of Hadoop related to HDFS:

- Hadoop supports shell-like commands to interact with HDFS directly.
- The NameNodes and DataNodes have been built in web servers that make it easy to check the current status of the cluster.
- Each of NameNodes and DataNodes runs an internal web server in order to display basic information about the current status of the cluster

3. Scientific Data Experiment Framework

We describe an overall framework to help build a MapReduce application, which supports to transform data in the field of scientific data experiments and its main considerations. The framework makes sure to seamlessly integrate pre-processing and post-processing phases with data analysis application. The designed framework for scientific data experiments consists of three entities: (1) scientific experimental workspace, (2) science data farm, and (3) domain data repository.

In the scientific experimental workspace, a researcher who has a plan of a scientific data experiment defines a workflow of this experiment that consists of processes with a set of activities. Each activity corresponds to a single process block that can be linked to another process block if control or data dependence exists between them. An activity might be something as straightforward as a data transformation that extracts specific values in a raw dataset. It can ease a researcher to use large-scale back-end computational resources such as HPC and cloud computing service. In a scientific data experiment, a workflow can be helpful to organize tasks of the data experiment. When a researcher does a scientific data experiment like simulation, other researcher working in a different domain may clearly define variables that can be ambiguous. In defining phases of a scientific data experiment, ontology is referred to resolve conflict of variables' unit and usage as well as ambiguity of them.

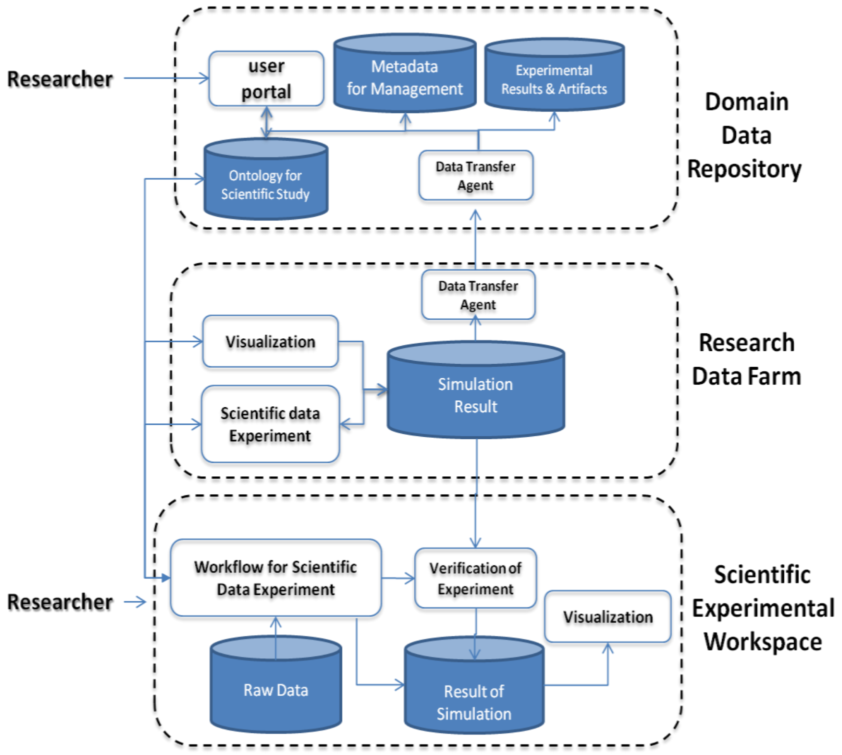

Figure 3. The Overall Framework for Scientific Data Experiments

According to the workflow defined in the scientific experimental workspace, a scientific data experiment is progressed on the scientific research data farm. Then, the intermediate and final data are moved to the domain data repository. In the domain data repository, researchers could share data, expertise and knowledge by user portal. The portal is designed to aggregate multiple information sources and applications to provide its users with uniform, seamless, and personalized access. Figure 3 shows the overall framework of scientific data experiments.

In the post-processing phase, the dataset generated during a simulation is typically validated, subjected to some initial processing and formatting and annotated with appropriated auxiliary information (i.e., metadata) to form a data collection. The type of post-processing varies with the application such as applying scientific codes to interpret the data in fusion applications, or generating multi-resolution representations of the data for visualization. In some cases the amount of data generated in the post-processing phase is equal to or greater than the original data, especially if indexes are built to be used in the analysis phase. Post-processing of the data can be done at multiple computational sites. Thus, large subsets of the original data can be moved to those computational sites and can be replicated at multiple locations. Different combinations of data from one or more data collections must be accessed and processed to produce the desired result. Unfortunately in many areas, data transfer bandwidths are growing at a much slower pace, making it extremely hard for scientists to download these rapidly growing datasets. Hence, data reduction is necessary to efficiently move a subset in the dataset. A typical workflow for scientific data experiments in the application structure is shown in Figure 4.

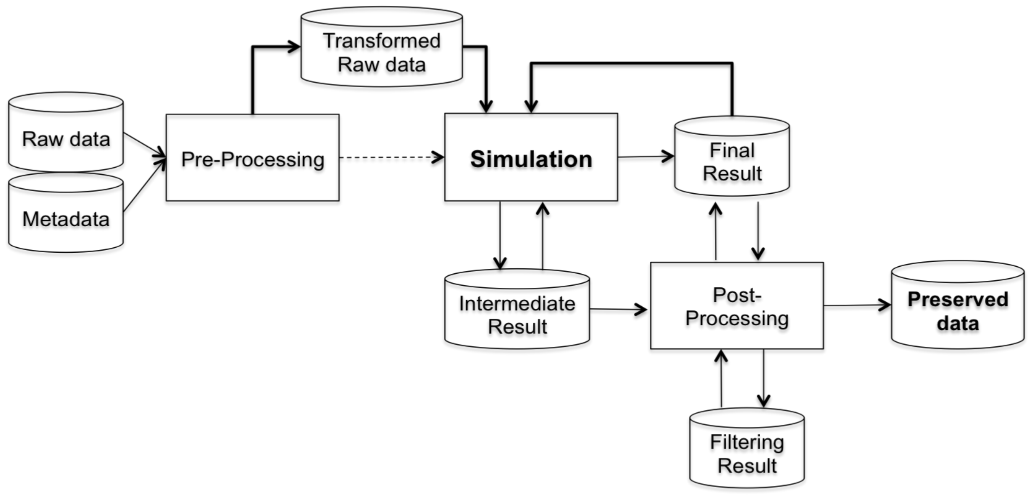

Figure 4. Basic Workflow for Scientific Data Experiments

4. Examples of Scientific Data Experiment

In this section, we describe a scientific application to evaluate the proposed framework, in which data transformation is applied to extract a set of data with specific variables. The data transformation in the framework is associated with a pre-processing and a post-processing phase.

4.1 Data Reduction of Dataset

Network common file (netCDF) is used to provide a common data access method for atmospheric science applications to manipulate a variety of data types that encompass single-point observations, time series, regularly spaced grids, and satellite or radar images [11]. Atmospheric science communication can access and transfer datasets encoded in netCDF that is an interface for array-oriented data access and a library that provides an implementation of the interface. The netCDF library also defines a machine-independent format to represent scientific data. Many methods have been developed for scientific data analysis. As one of such tools, NCO has been designed to execute common statistical operations over data in the netCDF file format [14]. However, the scalability of NCO is limited by the size of processing data.

The data generation phase may consist of running large simulations that require the full use of computing resources for many hours. During the phase, the main concern is to store the resulting data fast enough to avoid slowing the computer simulation. Datasets written in the netCDF format may also need to be moved to archival storage at the same rate to make room for data from the next simulation run. For example, climate models generate terabytes of data per simulation run, and increases in simulation resolution will greatly increase data storage requirements [9, 10, 11]. Even a conservative increase in the granularity of the model meshes will generate a factor of 16 more data. In the post-processing phase, the dataset generated during a simulation run is typically validated, subjected to some initial processing and formatting and annotated with appropriated auxiliary information. Moving such datasets is often prohibitive; rather a subset selection and filtering of the data need to be performed at the simulation site, and only the results are sent to the scientist for further analysis. The netCDF in Fig. 5 represents an output of the ensemble based climate simulation and the reduced netCDF represents a filtered output with specific variables in netCDF. Figure 5 shows the process of data reduction in a dataset written in a netCDF as the result of a simulation. This process is used to generate an input of statistical data analysis such as MapReduce.

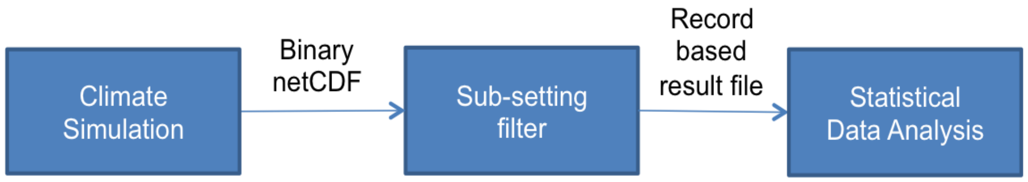

Figure 5. Basic Workflow of Data Reduction Process for Scientific Data Experiments

The graph in Figure 6 plots the access time of the sub-setting in the data reduction process. After sub-setting of five variables in the original netCDF file with 79 variables, the access time of one variable reduces up to 276 % in the part of the graph.

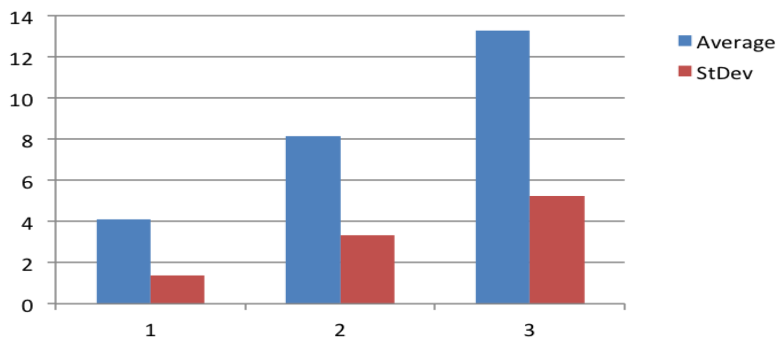

Figure 6. Speedup over Sub-setting of netCDF

4.2 Data Transformation for MapReduce Application

To apply data transformation of raw data from a dataset to the pre-processing of a scientific data experiment, we construct a MapReduce application that is used to handle a dataset collected from the National Climatic Data Center (NCDC, http://www.ncdc.noaa.gov/). The data is stored using the line-oriented ASCII format, in which each line is related to a separate record. The format supports a rich set of meteorological elements. Each record based on the state of the US maintains information about tornado outbreak from 1950 to 1990. It is organized by date with some columns including sequence number, state number, date, weather event, event-remarks, and the number of states involved. We conducted an experimental study to apply data transformation as a part of scientific data experiment to the designed framework. Hadoop v0.21, a MapReduce platform, is used to the experimental study.

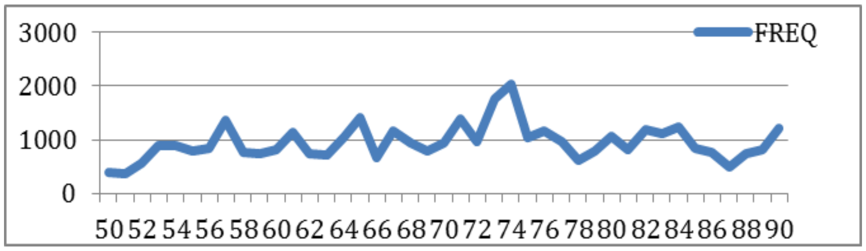

Figure 7. The Result of MapReduce Application Represents the Occurrence of Tornado with Regard to Year from 50 to 90

In the application we count the number of occurrences of tornado outbreak with regard to year and the state in the US in the dataset. Input key-value pairs take the form of (offset, line) pairs stored on the distributed file system, where the former is a unique identifier for the record, and the later is the record of the data itself. Each line represents a record of data associated with tornado outbreak. The input to the map function is a partition of the dataset represented by a multi-dimensional data to process. The map function extracts relevant data, which are associated with the specific variables in a record, by a filter function. The filter function in Map( ) is designed to extract an intensity value of a tornado from an input line. Reduce( ) groups the intermediate values with the key.

Figure 7 shows the result of the application that represents the number of tornado occurrence. Figure 8 shows the result of MapReduce associated with the partial sum of each of intensity of tornado outbreaks. From Figures 7 and 8, we can easily see that the frequency of tornado does not change, while the intensity is relatively increasing. Note that any detailed analysis to find more scientific meanings is beyond this paper's scope.

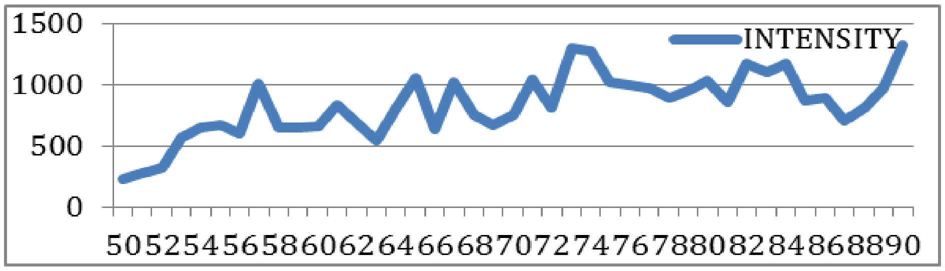

Figure 8. The Result of the MapReduce Application Represents the Intensity of Tornado Outbreaks

5. Conclusion

New scientific methods to analyze and organize data are required to handle large-scale datasets. Hence, these methods need to support effective infrastructure composed of computing resources that are used for pre-processing and post-processing data. We described the result of the empirical studies to handle dataset associated with a scientific data experiment which supports data reduction and data transformation, which are an essential phase to handling large scale data in scientific data experiments. This approach introduces a new way to process raw data by Hadoop, a MapReduce framework in a parallel manner. The performance of the designed MapReduce application was confirmed by summing up the variables such as tornado occurrence and its intensity.

References

[1] J. Gray, D. Liu, M. Nieto-Santisteban, A. Szalay, G. Heber and D. DeWitt, "Scientific data management in the coming decade", SIGMOD Rec., vol. 34, **(2005)**, pp. 34-41.
[2] Y. Han, "Bioworks: A Workflow for Automation of Bioinformatics Analysis Processes", International Journal of Bio-Science and Bio-Technology, vol. 3, no. 4, **(2011)**, pp. 59-68.
[3] J. Dean and S. Ghemawat, "MapReduce: A Flexible Data Processing Tool", CACM, vol. 53, **(2010)**, pp. 72-77.
[4] K. Morton, A. Friesen, M. Balazinska and D. Grossman, "Estimating the Progress of MapReduce Pipelines", IEEE 26th International Conference on Data Engineering (ICDE), **(2010)**, pp. 681 - 684, Long Beach, CA.
[5] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters", CACM, vol. 51, **(2008)**, pp. 107-113.
[6] F. Wang, J. Qiu, J. Yang, B. Dong, X. Li and Y. Li, "Hadoop high availability through metadata replication", Proceeding of the first International Workshop on Cloud data management, Hong Kong, China, **(2009)**.
[7] PolarGrid Project, http://www.PolarGrid.org/PolarGrid/index.php/Main_Page.
[8] I. Foster and C. Kesselman, "The Grid 2: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, **(2004)**.
[9] http://gcmd.nasa.gov/records/UCAR_WACCM.html.
[10] http://www.cesm.ucar.edu/models/atm-cam/.
[11] Ncar, http://www.unidata.ucar.edu/software/netcdf/.
[12] MPI. MPI (Message Passing Interface), http://www-unix.mcs.anl.gov/mpi/.
[13] Apache, http://hadoop.apache.org/.
[14] netCDF operator(NCO), http://nco.sourceforge.net/.

Authors

Yunhee Kang

He received the B.E. and M.S. degrees from Dongguk University, Seoul, Korea, in 1991, and 1993, respectively, and Ph.D. degree from Korea University, Seoul, Korea, in 2002, all in computer engineering.

He is currently an Assistant Professor with the Division of information and commutation, Baekseok University, Cheonan, Korea. His primary research interests lie broadly in distributed systems including fault-tolerance system, cloud computing and grid computing.

Heeyoul Choi

He received his B.S and M.S. degrees in Computer Science from Pohang University of Science and Technology, Pohang, Korea, in 2002 and 2005, and his Ph.D. degree from Texas A&M University in Computer Science 2010. He was a post-doc researcher in Indiana University, Indiana from 2010 to 2011. He is currently a research staff member at Samsung Advanced Institute of Technology, Samsung Electronics, since 2011. His research interests cover machine learning, data mining, pattern recognition, computational neuroscience, and cognitive science.