Enabling BigData Platform for MapReduce Applications in the Science Cloud

*1Yunhee Kang
1*Division of Information and Communication, Baekseok University, Cheonan, Korea 330-704, yhkang@bu.ac.kr*

Abstract

Scientific data experiments based on simulations create vast data stores that require new scientific methods to analyze and organize the data. The arrival of BigData science in the last two decades constitutes another scientific revolution. To handle BigData issues, cloud computing has gained significant traction in recent years. MapReduce is a practical and attractive programming model for parallel data processing in high-performance clusters virtualized of the cloud. Especially open source based MapReduce implementations are located as de facto standards in the science cloud projects. In this article we introduce research projects including LHC, PolarGrid and FutureGrid. We also describe software platforms including MapReduce framework and its extensions used for handling BigData problems in their projects

Keywords*: BigData, MapReduce Application, Science Cloud, software platforms, PolarGrid , FutureGrid*

1. Introduction

According to the growth of the number of mobile devices and the leverage of data in use of SNS(Social Network Service) based those devices, the attributes and challenges of big data have been addressed. Gartner highlighted that BigData is the top 10 technologies and trends that will be strategic for most organizations in 2013. Nowadays organizations have been generating massive amounts of data that are too large and too unmanageable to handle in a single system. Hence we are in an era of data deluge, where industry, academia, and governments are accumulating data at an unprecedentedly high speed. In addition the growing number of data-intensive applications is recognized and has lead to a wide range of approaches and solutions for data distribution, management and processing. For example scientific data experiments based on simulations create vast data stores that require new scientific methods to analyze and organize the data. The arrival of BigData science in the last two decades constitutes another scientific revolution [1-4].

In the mid 1990s, the term Grid was coined to describe technologies that would allow consumers to obtain computing power on demand [5]. Researchers subsequently developed these ideas in many exciting ways, producing, for example, large-scale federated systems that provide not just computing power, but also data and software, on demand. Cloud computing has also gained significant traction in recent years [10, 11]. It is a system architecture model for computing on architecture. It is a paradigm shift from the previous client-server architecture. Resources, software and information reside on this cloud and are shared by users [1, 6, 9, 19]. It is available to them only on demand.

The computing age of sharp growth and increasing data diversification is a major challenge for doing research. Our research is one of the efforts to apply cloud-based dynamic resource provisioning technology to Grid applications, in order to enable the scientific experiments in good performance under Grid environment [8, 9, 19]. The special responsibility of the study is to optimize the Grid platform utilizing virtualized resources, and to show how to provision the environment efficiently. To solve the scientific problem is moving forward a data intensive way from theoretical experiments. Hence Grid computing based on science cloud is considered to handle this problem. Many scientific applications require processes for handling data that no longer fit on a single cost-effective computer [8]. Besides, scientific data experiments such as simulations are

* Received: Jan. 18, 2014, Revised Jan. 30 Accepted: Mar. 25, 2014

creating vast data stores that require new scientific methods to analyze and organize the data. Those data experiments for handling scientific problems require the analysis of large amounts of data. Hence these challenges in solving problems need to be supported effective infrastructure composed of computing resources that are used for pre-processing and post-processing data as well as analyzing data [8].

There are two primary paradigms to handle data: the scientific high-performance (HPC) and the MapReduce based Hadoop paradigms, which we believe reflects and captures the dominant historical, technical and social forces that have shaped the landscape of Big Data analytics [17, 20].

In this article, we introduce research projects including LHC [27], PolarGrid [6] and FutureGrid [21]. We also describe software platforms including MapReduce framework and its extensions are used for handling Big Data problems in their projects. Especially we observe that open source based MapReduce implementations are located as de facto standards in the science cloud projects.

2. Evolution of scientific researches using Big Data

Modern theoretical science originated with Isaac Newton in the 17th century. After high-performance computers were developed in the latter half of the 20th century, Nobel Prize winner Ken Wilson identified computation and simulation as a third paradigm for scientific exploration. Detailed computer simulations capable of solving equations on a massive scale allowed scientists to explore fields of inquiry that were inaccessible by experiment or theory, such as climate modeling or galaxy formation. The fourth paradigm also involves powerful computers. But instead of developing programs based on known rules, scientists began with the data. They directed programs to mine enormous databases looking for relationships and correlations, in essence using the programs to discover the rules [12].

A paradigm shift is taking place to data driven research from hypothesis-driven research. The increasing ability to generate vast quantities of data presents both great opportunities and technical challenges for scientific discovery. It has become an important part of scientific discovery in an increasing number of disciplines. Researchers conduct experiments by analyzing the curated data collected from measurements and computational models. Even in the "small data" sciences, you see people collecting information and then having to put a lot more energy into the analysis of the information than they have done in getting the information in the first place [12].

In this new world where science is governed by computers and dominated by our computing technologies, scientists in this new era have to be very good in their own discipline. They have to understand computer science, computational thinking and they still have to know a lot about science in general. It will open a new era of computational research. The researchers have identified nearly two dozen research groups within Johns Hopkins alone that currently are struggling with data problems totaling to three petabytes or more.

Cyberinfrastructure (CI) enables scientific discovery by advancing new data intensive and simulation research paradigms. Emerging network, hardware, and software technologies continue to promise even greater opportunity for CI-based scientific discovery. Realization of this promise requires a powerful testbed that supports development and testing throughout the stack from networking to application code.

In verity of researches such as climate estimation, neural science, and astrophysics, the research which used to be done in the simulation-oriented way is being changed to a data-oriented way. Furthermore, in order to utilize large-scale data Grid technology expansively in diverse application areas, it is necessary to properly allocate the computing resources needed for large data storage capacity and long duration computation depending on the kinds of data experiments. So, feasibility analysis should be done in advance to figure out the possibility and effectiveness of the cloud-based test-bed that is strongly recommended as an efficient way of supporting computing resources.

3. Related Works: Science Projects based on BigData

3.1. Large Hadron Collider

The main goal of High Energy Physics (HEP) experiments is to process massive data generated by the Large Hadron Collider (LHC) as shown in Figure 1. The LHC at European Council for Nuclear Research (CERN) is expected to generate around 50 T Bytes of raw data per day. The LHC is expected to generate 10s of Petabytes of data annually after reducing the datasets via multiple layers of filtrations. The Worldwide LHC Computing Grid (WLCG) project as a data grid based on the community of HEP researchers is a global alliance of more than 150 computing centers that are spread out in the world, associated with national and international grid infrastructures [27].

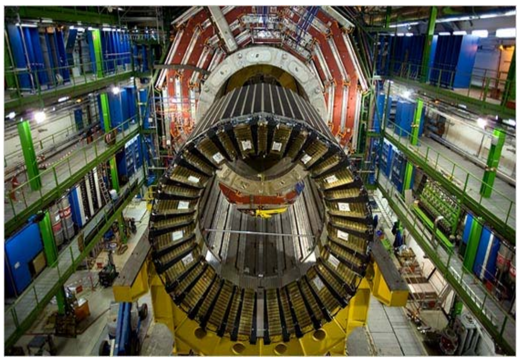

Figure 1. LHC at CERN

HEP applications require many computational resources and a large number of software dependencies on the diverse operating versions. High computational demands can be solved through the use of grid computing technologies. However the OS and software present at the resources that cannot be utilized. In the cloud computing environment with virtualized computing resources, the CERN VM, a software developed by a team of scientists at CERN, is used for HEP applications.

3.2. PolarGrid

Polar Grid is an NSF funded partnership of Indiana University and Elizabeth City State University to acquire and deploy the computing infrastructure needed to investigate the urgent problems in glacial melting [24]. It has done the aggregation of data acquired from radar instruments, which is the depth of ice sheets, at a based station in a polar grid research, to measure data after removing noise of those data by using large scale computing resources [6]. Figure 2 shows the sensor deployed in the polar region that is used for measuring the height of ice sheets.

Figure 2. Data sensor for measuring the height of ice sheets in polar area

PolarGrid cyberinfrastructure consists of ruggedized laptops and clusters deployed in the field in the polar region. Figure 3 shows the cluster installed at the base station at the polar region.

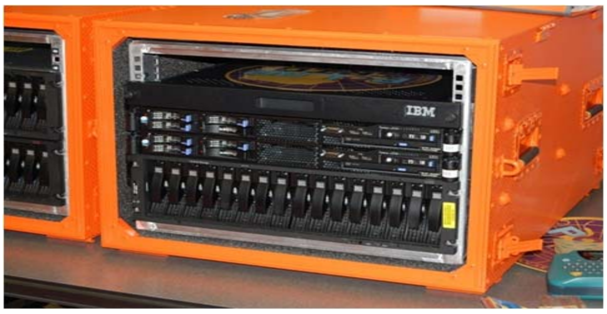

Figure 3. Cluster installed at the base station at polar area

Figure 4 shows the fast growth of data collected in the polar area since 2008. Finally the result of the project is used to predict the sea level of the US bay area and to analyze the effects of global warming related with the loss of ice sheets in the polar region. Researchers in other area are a portal can share and reuse those results. This will allow the scientists to identify problems with data collection and adjust experiments as necessary, to ensure that each expedition yields the highest possible quality of data. The deployed equipment also provides ample data storage and backup to prevent loss and damage of irreplaceable data.

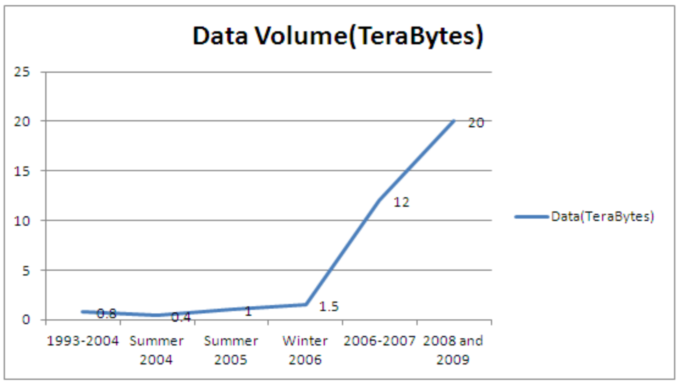

Figure 4. The growth of data collected in the polar area

To visualize the height of ice sheets from data collected, The Hidden Markov method is used to remove noise from them in the post-processing by HPC based computing resources [7]. Figure 5 shows the image about the height of ice sheet that is served from the portal of PolarGrid that is interacted with Google calendar, Google map, Facebook and Twister. The portal is a platform that gives researchers shared information and the results of experiments between them.

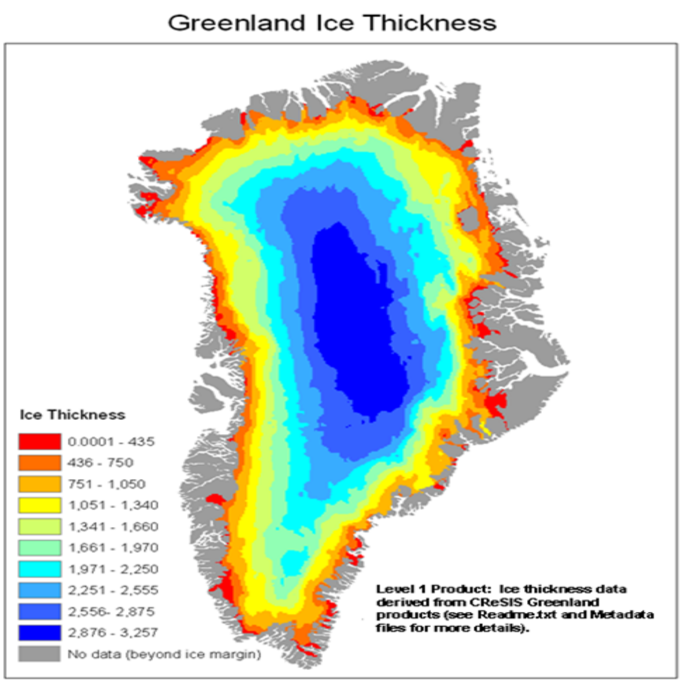

Figure 5. Image about thickness of ice sheets from GIS service provided by the portal of PolarGrid

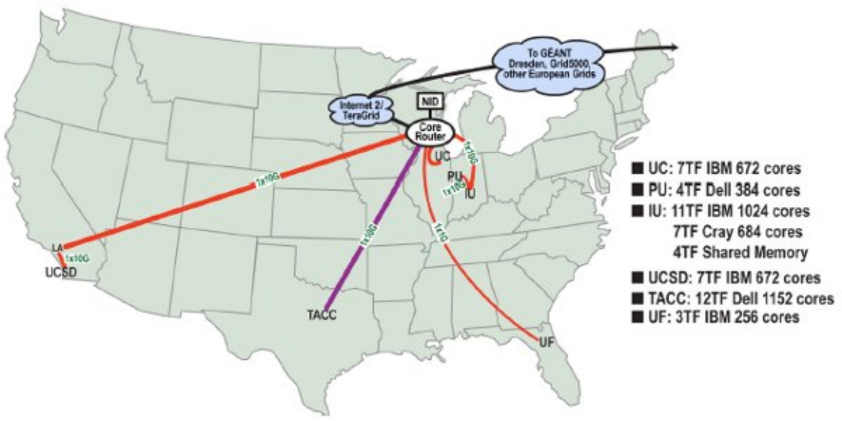

Figure 6. FutureGrid Connectivity

3.3. FutureGrid

FutureGrid is a project led by Indiana University and funded by the National Science Foundation (NSF). The main goal of FutureGrid is to build a distributed testbed for developing research applications and middleware. It applies virtualization technology to allow the testbed to support diverse operating systems. A machine image is used as a template when a machine instance represented as a virtual machine is an actual instantiation of the template. In the cloud FuturGrid, virtualization is used to reduce the actual number of physical servers and cost; more importantly, virtualization is used to achieve efficient CPU utilization of a physical machine. It has been offering a flexible reconfigurable testbed based on dynamically provisioning software to support deploying a specific image to a variety of environments composed of virtual machines [21]. Diverse configurations in the testbed have different operating systems and tools. The project provides a capability that makes it possible for scientific domain researchers to handle the problem with grand challenges aimed at reducing burden and optimizing performance in computer engineering related to the use and security of grids and clouds. The advantage of dynamic provisioning is maximizing utilization of a set of resources in an easy way to provide a user-defined environment to any user as a consumer of the resource. Figure 6 shows the data centers that are connected by FutureGrid.

FutureGrid with a cloud stack acts as a main part of resource providers in a science cloud. The demanding requirements in the FutureGrid have led to the development of a new programming model like MapReduce [18, 19]. A MapReduce application is deployed at the provider's computing infrastructure and its runtime supplies the developers a programming environment with a set of well-defined APIs that is referred to as PaaS. The MapReduce model enables large datasets to be divided and allocated to available computing nodes where the data can be processed locally, avoiding network-transfer delays. Cloud computing is good to run MapReduce application.

4. MapReduce Platform

4.1. Background of the MapReduce Programming Model

With the diversity of parallel applications and the need to process large volumes of data, we argue that simpler and data centered programming models like MapReduce can expand its usability to more classes of parallel applications.

Classic parallel applications using message-passing runtimes, such as MPI and PVM, utilizes a rich set of communication and synchronization constructs. Further the parallel algorithms running on these runtimes improve their performance by using a diverse set of communication constructs depending on topology. But the feature provided by a runtime library is a low-level primitive. So it involves a steep learning curve and potentially low programmer productivity.

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, and the framework takes care of the burden the programmer has to deal with lower-level mechanisms to control the data flow [13, 15]. The MapReduce programming model has introduced a great deal of interest because of its simplicity when writing a parallel program without low-level primitives.

By contrast MapReduce supports simple communication topologies and synchronization constructs. Simplicity of a MapReduce programming model supersedes the additional overheads. MapReduce is a programming framework popularized by Google and is used to simplify data processing across massive data sets [15].

4.2. MapReduce Programming Model

MapReduce, introduced by Google, is one of the most leading programming models for developing applications in the cloud computing environment [13, 15]. It is a kind of data parallel languages aimed at loosely coupled computations that execute over given data sets. This requires dividing the workload across a large number of machines. 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: mapping and reducing. These two functions have the following signatures:

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

The map function processes the input pairs (key1, value1) returning some other intermediary pair (key2, value2). Then the intermediary pairs are grouped together according to their keys. After that, each group will be processed by the reduce function which will output some new pairs of the form (key3, value3). When a MapReduce application is running with a MapReduce library, all map operations can be executed independently. In addition, the performance of reduce operation strongly depends on the outputs generated by any number of map operations. All reduce operations can also be executed independently. It is important that the map operation inputs on one <key, value> pair at a time. The map operations can manipulate keys arbitrarily, but the reducer operations cannot change the keys at all. When implementing a MapReduce program, the programmer has to write two functions: mapping() and reducing(). The function map() processes bulks of input data to produce intermediate results, and reduce() combines the intermediate results to produce the final output.

4.3. Hadoop

Hadoop [16] is an open source based on the MapReduce framework for running applications on large clusters built of commodity hardware from Apache. The Hadoop framework transparently provides applications with both reliability and data motion. It has become the de facto platform for large-scale data analysis in commercial applications, and increasingly so in scientific applications. However, Hadoop's byte stream data model causes inefficiencies when used to process scientific data that is commonly stored in highly-structured, array-based binary file formats. Hadoop is designed for cheap commodity hardware, co-places compute and data on the same node and is highly optimized for sequential reads workloads. With the uptake of Hadoop in the commercial space, scientific applications and infrastructure providers started to evaluate Hadoop for their purposes.

Hadoop implements a MapReduce application that 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 compute nodes, providing 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 compute nodes throughout a cluster to enable reliable, extremely rapid computations. Hadoop supports shell-like commands to interact with HDFS directly. The NameNode and DataNode have built in web servers that make it easy to check current status of the cluster. The NameNode and DataNode respectively run an internal web server in order to display basic information about the current status of the cluster. It uses HDFS as shared storage, enabling data to be shared among distributed processes using files. In the present implementation of Hadoop, latency for reading a random chunk of data from HDFS can range from tens of milliseconds in the best case to possibly hundreds of milliseconds in the worst case.

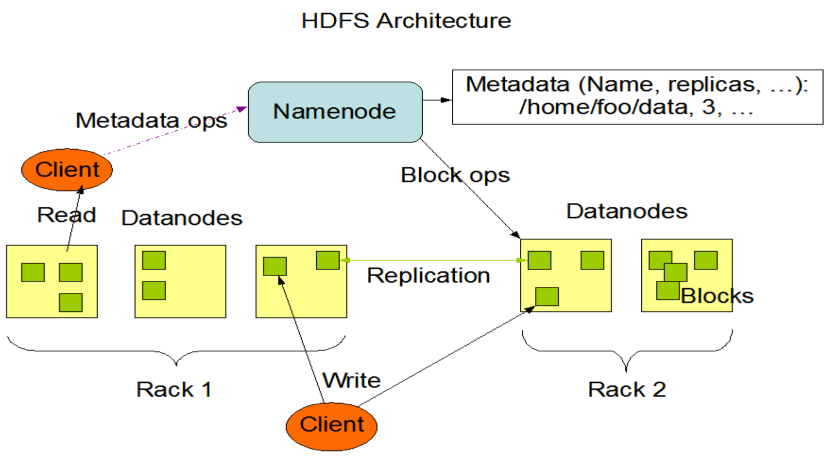

Figure 8. HDFS Architecture of Hadoop

Basically, MapReduce is designed for batch-oriented processing computations that involve sequential reads of large files. To create a new file and write data to HDFS, the MapReduce application first contacts the NameNode, which updates the file namespace after checking permissions and making sure the file doesn't already exist. The HDFS implementation has the master-slave architecture: the master process running on the NameNode manages the global name space and controls operations on files, which 가 아니라 while 아닌가요 a slave process running on the DataNode performs operations on blocks stored locally, following instructions from the NameNode. Figure 8 shows the overall architecture of HDFS in Hadoop.

4.4. Pros and Cons: The characteristics of MapReduce Applications

A MapReduce application can impose a heavy network load. This is an especially important consideration when applications are running on large-scale Hadoop clusters. The two phases of a MapReduce job has two candidates (i.e., shuffling and reduce output) stressing network interconnects in Hadoop clusters. During the shuffle phase, each reduce task contacts every other node in the cluster to collect intermediate files. During the reduce output phase, the final results of the entire job are written to HDFS.

Map-only applications are bags of independent tasks and clearly are suitable for clouds. The parallel case includes not only LHC and similar science analysis but also support of the IoT where each of the widely distributed devices is served by the cloud. Various other approaches for supporting data-intensive applications on the HPC structures emerged.

In contrast, data-intensive workloads can be decomposed into fine-grained, loosely-coupled parallel tasks. For example, MapReduce implementations for HPC have been proposed: MPI-based MapReduce implementations can utilize HPC features. MapReduce extensions have been proposed. Combining the MapReduce framework with the cloud provides a number of unique advantages. It is particularly appealing for organizations that need to analyze large amounts of data without having to acquire and manage large cluster resources. The user does not need to own the cluster resources required to run the job. MapReduce is arguably the most successful parallelization framework used in datacenters comprising commodity computers. The Hadoop has seen active development activities and increasing adoption. It is designed to scale up from a single server to thousands of machines, offering local computation and storage.

Unfortunately, a naive approach to processing scientific data with MapReduce is difficult. Typically the input to a MapReduce computation is a file stored as blocks in a distributed file system, and MapReduce processes each block of the input file in parallel. Thus the difficulty of processing scientific data with MapReduce is manifested as a scalability limitation.

Today storage devices are built for the byte streams data model, thus high-level data models such as arrays must be translated onto low-level byte streams. As data storage and transfer approaches critical mass, more and more scientific fields are in the process of adapting the traditional research methodologies to be driven by large-scale data analysis. Furthermore, analysis may require a complex series of processing steps, each generating intermediate data products need to be stored, either temporarily or permanently, and made available for discovery and use by other analysis processes.

Most of these types of algorithms can be parallelized by applying the SPMD style to the main computations that are executed inside the iterations. Depending on the algorithm, there can be one or more SPMD steps inside the main iterative construct. However, the side effect free nature of the MapReduce programming model does not fit well for iterative MapReduce computations in which each map and reduce tasks are considered as atomic execution units with no state shared in between executions. Without this step, there would be no way to perform multiple iterations. MapReduce jobs have high startup costs. Basically MapReduce often create a straggler in the reduce phase. The highly uneven distribution of incoming edges to vertices produces significantly more work for some reduce tasks. Note that since these stragglers are caused by data skew, speculative execution cannot solve the problem. At each iteration, the algorithm must shuffle the data structure from mappers to the reducers. Since, in most cases, the data structure is static, the represents wasted effort.

To overcome these limitations, a number of extensions to MapReduce or alternative programming models have been proposed.

Main practical computing problems concern large graphs. Efficient processing of large graphs is challenging. Pregel implements the Bulk Synchronous Parallel model. It is similar in concept to MapReduce, but with a natural graph API and much more efficient support for iterative computations over the graph. The graph needs to be broken down into multiple partitions and stored in different places. But the traditional graph algorithm that assumes the whole graph can be resided in memory becomes invalid. On the other hand, by breaking the graph into different partitions, it can manipulate the graph in parallel to speed up the processing. Pregel computations consist of a sequence of iterations [25]. It is suitable for handling large-scale graph computing.

Twister is one of MapReduce implementations that is an enhanced MapReduce runtime with an extended programming model that supports iterative MapReduce computing efficiently [22]. In Twister, another extension of Hadoop designed for iteration, intermediate data are retained in memory if possible, thus greatly reducing iteration overhead [18, 19]. In addition it provides programming extensions to MapReduce with broadcast and scatter type for transferring data. It reads data from the local disks of the worker nodes and handles the intermediate data in the distributed memory of the worker nodes. As shown in Figure 9, all communication and data transfers are performed via a pub/sub broker network that is a distributed messaging infrastructure. But it is difficult to understand the behavior of a MapReduce application caused by hiding behavior in details associated with status (when running it) within its runtime. In addition the behavior of a MapReduce application also depends on its framework when it is running. Especially slow tasks are a major performance bottleneck in MapReduce systems. The characteristics of a MapReduce application are also strongly related with disk IO, CPU cycle and network bandwidth.

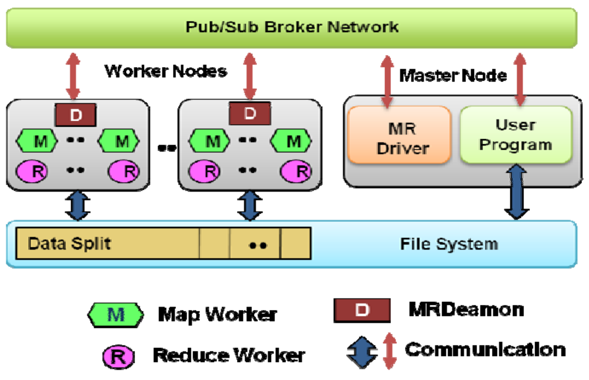

Figure 9. The overall architecture of Twister

One of the challenges is the shear scale: the data simply can't be moved for analysis. Clustering, classification, pattern mining and dimension reduction are some of the areas where many of the iterative algorithms are used. Therefore the data must be analyzed in situ and we must develop methods for extracting a smaller set of relevant data. In-situ MapReduce offers a pathway forward using commodity hardware that most universities and research centers face these days and the emerging power wall, which is soon to become the limiting factor in deploying high-performance computing [26].

5. Conclusion

The growth of the volume of data requires the amount of computing resources including much larger scales than a traditional computer cluster. Computing resources have been diverse from supercomputers to on-demand resources in cloud computing. This also makes writing parallel programs inevitable. But writing a distributed and parallel application using MPI is not easy caused of low-level primitive like synchronization and inter-process communication. The demanding requirements have led to the development of a new programming model and its implementations like MapReduce. The MapReduce paradigm offers a simple API for computation to a programmer. Performance of a MapReduce application is related with its characteristics.

New scientific methods to analyze and organize the 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 design of a framework to support data transformation, in which is an essential phase to handling a large scale of data in scientific data experiments. This approach introduces the way to process raw data by Hadoop, a MapReduce framework in a parallel manner. The designed MapReduce application is used to sum up the variables, such as tornado occurrence and its intensity.

But it is difficult to understand the behavior of MapReduce application caused by hiding behavior in details associated with status (when running it) within its runtime. In addition the behavior of a MapReduce application also depends on its framework when it is running. Especially slow tasks are a major performance bottleneck in MapReduce systems. The characteristics for execution of a MapReduce application is also strongly related with disk IO, CPU cycle and network bandwidth.

The article can be utilized for diverse computation application by means of Science Cloud platform that has been optimized through this research. In the future, it will also be useful in the composition of the scientific application environment that demands high computing resources as well as large data storage.

6. References

[1] P. Zikopoulos and C. Eaton, "Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data," *McGraw-Hill Osborne Media*, 1st ed., 2011.

[2] D. Agrawal, S. Das, and A. E. Abbadi, "Big Data and Cloud. Computing: New Wine or just New Bottles?," *PVLDB*, Vol. 3, No. 2, pp.1647–1648, 2010.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, 2009.

[4] Y. Han, "Bioworks: A Workflow for Automation of Bioinformatics Analysis Processes," *International Journal of Bio-Science and Bio-Technology*., Vol. 3, No. 4, pp. 59–68, 2011.

[5] I. Foster and C. Kesselman, "The Grid 2: Blueprint for a New Computing Infrastructure," *Morgan Kaufmann*, San Francisco, CA, USA, 2004.

[6] Z. Guo, R. Singh, and M. Pierce, "Building the PolarGrid Portal Using Web 2.0 and OpenSocial," in Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Portland, Oregon, USA, Nov. 2009.

[7] L. Hayden, G. C. Fox, and A. Adade, "Implementing Cyberinfrastructure in Support of Greenland and Antarctic SAR Data Sets," in Proc. of the International Conference of the African Association of Remote Sensing of the Environment (AARSE), Accra, Ghana, Oct. 2008.

[8] Y. Kang and H. Choi, "An Empirical Study for Handling Scientific Datasets," *International Journal of Grid and Distributed Computing*, Vol. 5, No. 3, pp. 111–120, Sept. 2012.

[9] Y. Kang, "Construction of a MapReduce Application Running on Twister in Cloud Computing Environments, FutureGrid", *Journal of KIIT*, Vol. 9, No. 4, Apr. 2011.

[10] D. A. Patterson, "Technical Perspective: The Data Center is the Computer," *Communications of the ACM*, Vol. 51, Iss. 1, pp. 105–105, Jan. 2008.

[11] B. Hayes, "Cloud computing", *Communications of the ACM*, Vol. 51, No. 7, pp. 9–11, July 2008.

[12] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber, "Scientific data management in the coming decade," *SIGMOD* Rec. 34, 4, pp. 34–41, Dec. 2005.

[13] J. Dean and S. Ghemawat, "MapReduce: A Flexible Data Processing Tool," *Communications of the ACM*, Vol. 53, pp. 72–77, 2010.

[14] K. Morton, A. Friesen, M. Balazinska, and D. Grossman, "Estimating the Progress of MapReduce Pipelines," in Proc. of the IEEE 26th International Conference on Data Engineering (ICDE), Long Beach, CA, USA, Mar. 2010, pp. 681–684.

[15] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Communications of the ACM*, Vol. 51, pp. 107–113, Jan. 2008.

[16] Hadoop. [Online]. Available: http://hadoop.apache.org/

[17] F. Wang, J. Qiu, J. Yang, B. Dong, X. Li, and Y. Li, "Hadoop high availability through metadata replication", in Proc. of the 1st International Workshop on Cloud data management (CloudDB), Hong Kong, China, Nov. 2009, pp. 37–44.

[18] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, and G. Fox, "Twister: A Runtime for Iterative MapReduce," in Proc. of the 1st International Workshop on MapReduce and its Applications (MAPREDUCE), June 2010, pp. 810–818.

[19] Y. Kang and G. C. Fox, "Performance Evaluation of MapReduce Applications on Cloud Computing Environment", in Proc. of the Grid and Distributed Computing (GDC), Jeju, South Korea, Jan. 2011, pp. 77–86.

[20] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, "A comparison of approaches to large-scale data analysis," in Proc. of the ACM SIGMOD International Conference on Management of data (SIGMOD), Rhode Island, USA, June-July 2009, pp. 165–178.

[21] FutureGrid. *FututrGrid Portal* [Online]. Available: https://portal.futuregrid.org/

[22] Y. Kang, "Construction of a MapReduce Application Running on Twister in Cloud Computing Environments, FutureGrid", *Journal of KIIT*, Vol. 9 No. 4, pp. 147–154, April 2011.

[23] M. Mahjoub, A. Mdhaffar, R. B. Halima, and M. Jmaiel, "A Comparative Study of the Current Cloud Computing Technologies and Offers", in Proc. of the 1st IEEE Symposium on Network Cloud Computing and Applications (NCCA), Toulouse, France, Nov. 2011, pp. 131–134.

[24] PolarGrid. *PolarGrid Portal* [Online]. Available: http://www.polargrid.org/polargrid

[25] G. Malewicz, M. H. Austern, A. J. C Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing," in Proc. of the ACM International Conference on Management of data (SIGMOD), New York, NY, USA, June 2010, pp. 135–146.

[26] D. Logothetis, C. Trezzo, K. C. Webb, and K. Yocum, "In-situ MapReduce for log processing," in Proc. of the USENIX conference on USENIX Annual Technical Conference (USENIXATC), Portland, Oregon, USA, 2011, pp. 9–9.

[27] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing," *ACM Comput. Survey,* Vol. 38, No. 1, Article 3, June 2006.