SciencePG
Science Publishing Group

Simplified Data Processing for Large Cluster: A MapReduce and Hadoop Based Study

Abdiaziz Omar Hassan[*], Abdulkadir Abdulahi Hasan

College of Mathematics and Big Data, Anhui University of Science and Technology, Huainan, China

Email address:
engmahir00@gmail.com (A. O. Hassan)
[*]Corresponding author

Abstract: With the drastic development of computing technologies, there is an ever-increasing trend in the growth of data. Data scientists are overwhelmed with such a large and ever-increasing amount of data, as this now requires more processing channels. The big concern arising here for large-scale data is to provide support for the decision making process. Here in this study, the MapReduce programming model is applied, an associated implementation introduced by Google. This programming model involves the computation of two functions; Map and Reduce. The MapReduce libraries automatically parallelize the computation and handle complex tasks including big data distribution, loads and fault tolerance. This MapReduce implementation with the source formation of Google and the open-source mechanism, Hadoop has an objective of handling computation of large clusters of commodities. Our implication of MapReduce and Hadoop framework is aimed at discussing terabytes and petabytes of storage with thousands of machines parallel to every machine and process at identical times. This way, large processing and manipulation of big data are maintained with effective result orientations. This study will show up the basics of MapReduce programming and open-source Hadoop structure application. The Hadoop system can speed up the handling of big data and respond very fast.

Keywords: Google MapReduce Processes, Hadoop, Parallel Data Processing, HDFS, Cloud Computing,
Large Cluster Data Processing

1. Introduction

With the introduction and advancement of technology and computerized innovation, the growth of data is unimaginable and unreachable. Data scientists and handlers are getting overwhelmed and frustrated with such a large and ever-increasing amount of data with its processing requirements ever-increasing and demanding more every time. With so large an ever-increasing data, there comes to some problems as well concerning its handling, processing, and management. These problems are faced by various fields in making use of this large scale, drawing meanings out of it, as well as, using it for decision making.

Data mining, data classification, handling, and processing are some of those technologies that can amend and draw new ways out of these large data sets. For many years in the past, this data mining technique with its pre-requisites is studied in all applicable scenarios; resulting it to be the phase of development of data mining methods and further their application to make them workable. Various hurdles in the wake of processing are faced by large-scale internet companies including Google, Yahoo, Facebook, LinkedIn, as well as, other bigger internet-solution providing companies that require processing a huge chunk of data not only in minimum timeframe but also keeping the cost-effective solution in an application.

Google had developed MapReduce and the Google File System, which is embracing to studied and investigated in this research study. Google has also built a database management system (DBMS) known as Big Table. This system can search millions of pages and return the results in milliseconds by employing some algorithms that work through the MapReduce system and Google File System [1].

In the recent past, MapReduce has made its place as an algorithm to handle computing paradigm and analysis of a large amount of data [2]. MapReduce has got fame while it

was made part of the Google database management system and Google file system. MapReduce could be employed for measurability and is purely a fault-tolerant data processing tool that can handle and process huge data along with lower-bound computing nodes [3].

Discussing how MapReduce works, a distributed file system (DFS) first categorizes data in multiple categories, and then data is presented as a pair containing key and values. The MapReduce framework performs its applications and function on a single machine where the data may be preprocessed before map functions or post-process the output of MapReduce function performed [4]. As Hadoop is applied, which is a famous open-source application of MapReduce to handle large datasets. It employs an already provided user-level filesystem to handle storage across the cluster [5]. This implication will provide you with a speedy output but less significant, yet giving you a reasonable speed as well as handling a larger dataset that tackles a large number of computing nodes and minimizes application time by 30% comparing with ordinary data mining techniques [6].

1.1. Programming Model and Application of MapReduce Function

The programming model indicates and includes defined sets of input pairs of key or value, and outlays output pairs of key and values. This MapReduce function has two outlays: one is Map and the other, Reduce. The map function considers the input pair and provides key/value pairs. These values and intermediate outputs are grouped by the MapReduce function and then passed further to the Reduce function. The Reduce function accepts these intermediate keys and merges their values to form a smaller set of values. Let us assume an example of counting the occurrence of each word in a large dataset, and appear to use it with the map and reduce functions. The code to do this counting of occurrence will be similar to the following code:

map (String key, String value):// key: document name// value: document contents for each wordw in value: EmitIntermediate(w, "1");

reduce (String key, Iterator values):// key: a word// values: a list of counts int result = 0; for each v in values: result += ParseInt (v); Emit(As String(result));

Herewith this example, the program in detail counts the occurrences of each word within input files specified on the command line.

#include "mapreduce/mapreduce.h" // User's map function class Word Counter: public Mapper {public: virtual void Map (const MapInput& input) {const string& text = input.value(); const int n = text.size(); for (int i = 0; i < n;) {// Skip past leading whitespace while ((i < n) && isspace(text[i])) i++; // Find word end int start = I; while ((i < n) &!isspace(text[i])) i++; if (start < I) Emit(text.substr(start,i-start),"1");}}};

REGISTER_MAPPER(WordCounter); // User's reduce function class Adder: public Reducer {virtual void Reduce (ReduceInput* input) {// Iterate over all entries with the// same key and add the values int64 value = 0; while (!input->done()) {value += StringToInt(input->value()); input->NextValue(); // Emit sum for input->key () Emit (IntToString(value));}};}

REGISTER_REDUCER(Adder); int main (int argc, char** argv) {ParseCommandLineFlags(argc, argv);

1.2. MapReduce Specification

spec;// Store list of input files into "spec" for (int i = 1; i < argc; i++) {MapReduceInput* input = spec.add_input(); input->set_format("text"); input->set_filepattern(argv[i]); input->set_mapper_class("WordCounter");} // Specify the output files: // /gfs/test/freq-00000-of-00100 // /gfs/test/freq-00001-of-00100 //
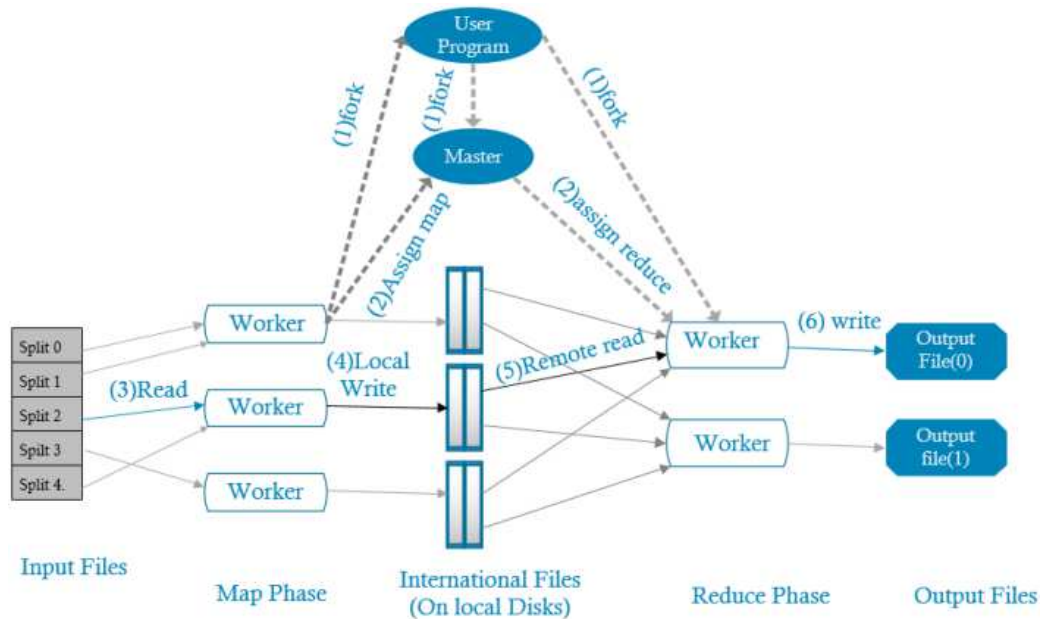

Figure 1. *Execution plan of the programming model.*

1.3. MapReduce Output

out = spec.output(); out->set_filebase("/gfs/test/freq"); out->set_num_tasks(100);   out->set_format("text");   out->set_reducer_class("Adder") // Optional: do partial sums within map // tasks to save network bandwidth out->set_combiner_class("Adder"); // Tuning parameters: use at most 2000 // machines and 100 MB of memory per task

spec.set_machines(2000); spec.set_map_megabytes(100); spec.set_reduce_megabytes(100); // Now run it

The above code was written in terms of string inputs and outputs, but the map and reduce functions have also associated types that are defined and assessed through the listing of variables and values.

map (k1,v1) → list(k2,v2) reduce (k2,list(v2)) → list(v2)

The map function will dig out the key from each recording and will forward the key with a matching record pair. On the other hand, the reduce function will give all pairs unchanged.
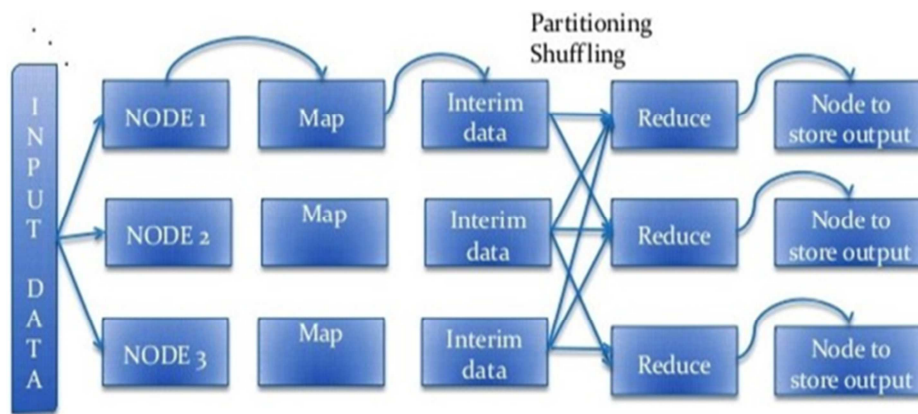
2. Related Works

Seema Maitrey with her fellow researchers has studied big data handling with a new technique under the name of "MapReduce: Simplified Data Analysis of Big Data".

This research study is focused on using the MapReduce technique that is based on cloud-based technologies. A famous application of cloud technology is Google, which works aligned with this technology and handles data and processes with care. They also discussed Hadoop that is used by companies other than Google, including Facebook, Yahoo, etc. The analytical processing of data using Hadoop and the application of MapReduce is verified and assessed with their research-based study [7].

Another researcher Jeffrey Dean with his fellow researchers has studied the MapReduce framework getting a lot of attention for the application on big data. They classified it as a programming model with implementation with the aim of processing and handling large datasets being responsive for a wide variety of real-world operations [8].

Figure 2. Anatomy of MapReduce function.

Richard M. Yoo and his fellows have studied Scalable MapReduce with a large-scale shared-memory system and talked about dynamic runtimes with simplifying parallel programming, as well as automatically detecting scenarios. They discussed how a multi-layered approach works along that work for the optimizations on the algorithm, implementation, and OS interaction defining and channelizing significant speedup improvements with 256 threads. They also identified the hurdles or roadblocks which are involved in limiting the scalability of runtimes on shared-memory systems [9].

Kyong Lee with his fellows had discussed Google's MapReduce technique that works for big data handling and processing more simply and smoothly together with the benefit of minimized cost. The main characteristic of this MapReduce model was that it able to process large data sets among others distributed among multiple nodes and multiple channels [10].

B Panda and his fellows had highlighted the MapReduce system and its applications with big data at an International

Conference. They highlighted the MapReduce mechanism being a proprietary system of Google. They also discussed the distributed computing being great to extend simplified with implications of Map and Reduce functions, providing the basics and insights of achieving the desired performance [11].

Jeffrey Dean with his fellows had discussed simplified data processing on large clusters with the MapReduce framework. They stated this being the subsidiary infrastructure of Google's MapReduce that allocates to a distributed file system and enables the algorithms to locate data and make it available. They termed it easy to use as with the opinion of programmers as more than ten thousand distinct MapReduce programs are on implementation internally at Google within the last four-year span [12].

Bayardo Panda and his fellows have discussed massively parallel learning with the application of the MapReduce framework. They highlighted combining the MapReduce programming technique with the distributed file system, being a way to achieve distributed computing objectives with data processing over thousands of computing nodes [11].

Jaliya Ekanayake and her fellows have discussed MapReduce for data-intensive scientific analysis. They discussed the MapReduce technique due to its application to large parallel data analyses. They discussed this with efficient parallel/concurrent algorithms meeting the scalability and performance requirements to handle and process scientific data [13].

Anam Alam and her fellows have discussed the Hadoop Architecture and Its Issues, together with their implication at an international conference. Hadoop is categorized as a distributed program or framework used to handle a large amount of data. Hadoop is usually used for data-intensive applications. With its extensive application, every social media site has made use of it [14].

R. Vijayakumari and her fellows have discussed the comparative analysis of Google File System and Hadoop Distributed File System. They discussed this distributed computing, parallel computing, grid computing, and other parameters including; Design Goals, Processes, Fire management, Scalability, Protection, Security, cache management replication, etc. to compare both these methods and their application of the file system [15].

3. Methodology

The methods used may not look familiar to a common audience. The first one is MapReduce which is in fact oriented to programmers, rather than business users. This has gained popularity due to its easy application, efficiency and ability to control "Big Data" in a timely manner. MapReduce framework with its application and programming model is discussed above. An example of occurrences is discussed and employed with the MapReduce framework.

3.1. Hadoop

Another process employed and utilized is Hadoop which is connected with Java implementation and Java application.

This should be used in two different ways. These are the outputs advantageous API streamed output and the other involving building of Hadoop apps with C++. Hadoop Distributed File System is a target file system especially to use with MapReduce programs. This best applicable to the small number of very large files. With the use of replication, data availability could be made possible within Hadoop Distributed File System (HDFS).

To process all of the files created by the mapping mechanism, the Reduce program get access to internode data. When this is executed, map and reduce, both programs will write it down to the local file system to avoid the burden over the HDFS system. HDFS can support multiple readers and one writer (MROW) approach. The indexing mechanism might not apply to HDFS, so, this would just be applied to read-only applications that only scan and read the contents of the file.

3.1.1. Hadoop Architecture

Hadoop Distributed File System stores data within its computing nodes, providing customized and high aggregate bandwidth across the entire cluster. This file system installation has different nodes plus one single name node, called the master node and various data nodes, called slave nodes. The name node has held responsible for the management of the file system namespace and controls the access to files by clients. The data nodes or slave nodes are distributed in a way that one data node is assigned per machine in the cluster, managing data while attaching it to the machines where they run. The name node has an operation execution scenario within the file system namespace and assigns those data blocks to data nodes. Those data nodes are there to handle read and write requests from clients and performing operations with the instruction provided [16].
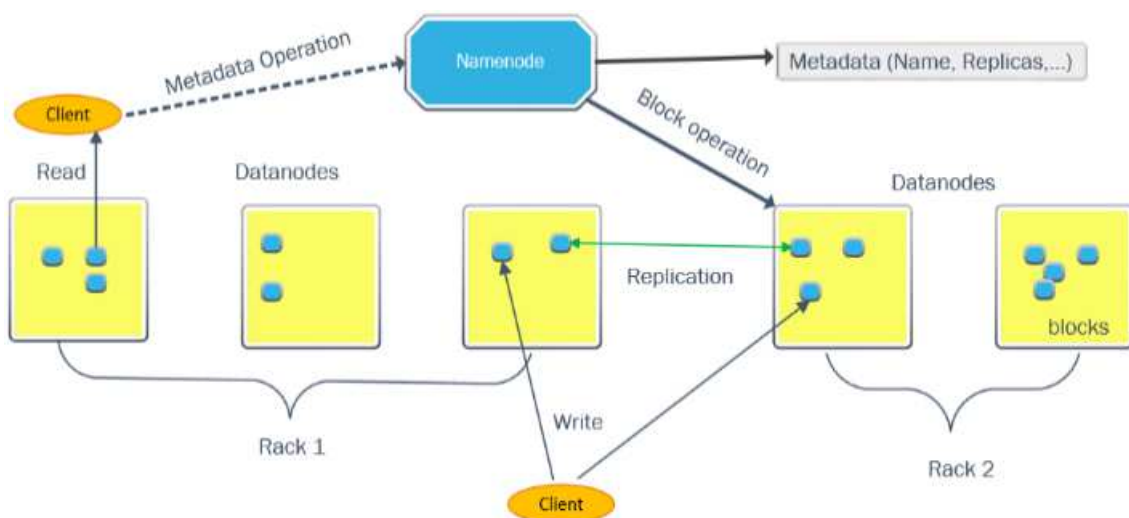

Figure 3. *Hadoop Architecture.*

Hadoop Distributed File system manipulates and handle data chunks and replicates these data chunks across the

server for performance keeping and mechanism, load-balancing and resiliency. The processing application of any

problem execution will specify the number of replicates of the file right when it is created, and this count or record can be changed any time after that. The name node has the ability to adopt different decisions concerning block replication.

3.1.2. Deploying Hadoop

Hadoop compiles in three different ways, the first one is a standalone mode, which is the default mode of Hadoop, running as a single Java process. The second one is Pseudo-distributed mode, which involves the configuration of Hadoop to run on a single machine, whereas, with different Hadoop processes, run divergent Java processes. The third one is fully disseminating or cluster mode, involving one

machine, as the name node and another, as the job tracker. There could be a secondary name node that might work for periodic handshaking with a name node for fault tolerance.

3.1.3. Replication Management

HDFS provides a reliable way to store huge data in a distributed environment as data blocks. The blocks are also replicated to provide fault tolerance. The default replication factor is 3 which is again configurable. So, as you can see in the figure below where each block is replicated three times and stored on different DataNodes (considering the default replication factor):
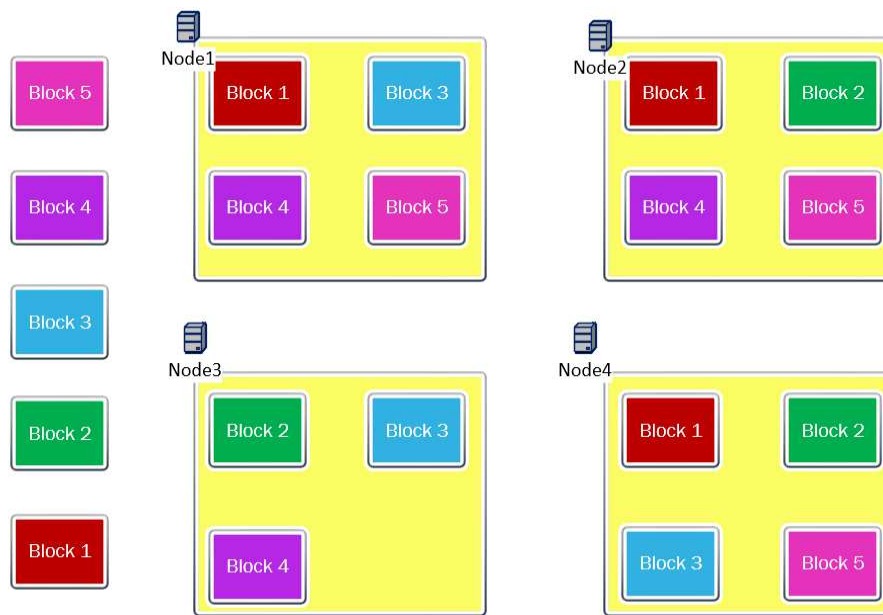

Figure 4. Block replication.

3.2. Hadoop Based Oozie Structure and Implementation

Apache Oozie manages all the tasks and makes them organized. So, this could be known as a scheduler for Hadoop. This mechanism provides workflow of dependent jobs that later on and helps to develop Directed Acyclic Graphs of workflows that allow jobs or tasks to run in parallel and sequentially in Hadoop.
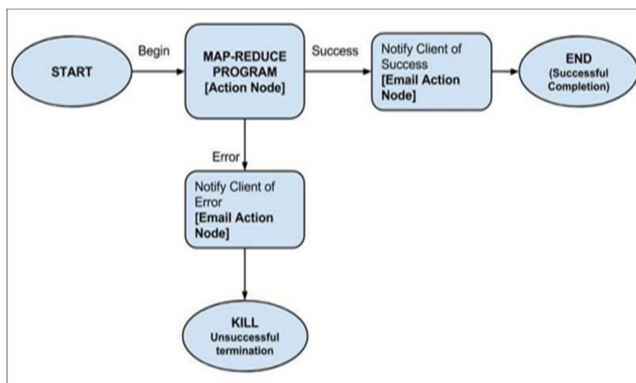

Figure 5. Oozie workflow chart.

This type of Oozie workflow works with both action nodes and control-flow nodes. An action node represents a workflow task like moving files into HDFS, running a MapReduce, or running a shell script of a program written in Java. While a control-flow node controls the execution of the task by allowing different action nodes and controlling control nodes.

4. Results and Discussions

Discussing results and discussions, big data and its requisite technologies can bring about significant changes and benefits to your business. But with the increased and widespread use of technologies, it might turn into a difficult task for your organization to manage, control and tackle a heterogeneous collection of data and get your desired outcomes.

To handle the growth of individual companies, certain aspects should be followed so that timely results could be attained from Big Data since effective use of Big Data, the modernization, and effectiveness for entire divisions and economies are to be attained. Therefore, you should know how to ensure the effectiveness of usage, management and

re-use of data sources, including public data to construct applications. There is a need to evaluate the best approach to use for filtering and analyzing the data. For the optimized processing, Hadoop with MapReduce could be employed. As we have used in this paper, with the basics of MapReduce programming and open-source Hadoop framework application. The Hadoop framework can speed up the processing of big data and respond very fast. The extensibility and simplicity of these frameworks will be the critical factors that make it a replenishing tool for big data handling, processing, and management.

5. Conclusion

MapReduce programming model is applied, an associated implementation introduced by Google. This programming model involves the computation of two gatherings; Map and Reduce.

Hadoop performance is made up of an ecosystem of tools and technologies that will requirement careful analysis and expertise to determine the suitable mapping of technologies to enable a smooth migration.

Hadoop is a highly scalable platform and is largely because of its ability that it stores and allocates large data sets across lots of servers. The servers used here are quite inexpensive and can operate in parallel. The processing power of the system can be improved with the addition of more servers.

Hadoop MapReduce programming model offers suppleness to process structure or unstructured data by several business organizations who can use the data and operate on different types of data. Thus, they can achieve a business value out of those meaningful and beneficial data for the business organizations for analysis.

References

[1]   G. Z. &. C. B. Jason R Swedlow, "Channeling the data deluge," *Nature methods,* vol. 8, p. 463–465, 2011.

[2]   J. Maitrey S, "An Integrated Approach for CURE Clustering using Map-Reduce Techniques," *In Proceedings of Elsevier,* vol. 2, 2013.

[3]   D. D, "MapReduce: A major step backwards," *The Database Column,* 2011.

[4]   Y. Kim and K. Shim, "Parallel Top-K Similarity Join Algorithms Using MapReduce," Arlington, VA, USA, 2012.

[5]   J. Shafer, S. Rixner and A. L. Cox, "The Hadoop distributed filesystem: Balancing portability and performance," White Plains, NY, USA, 2010.

[6]   S. M. CA Moturi, "Use of MapReduce for Data Mining and Data Optimization on a Web Portal," *International Journal of Computer,* vol. 56, no. 7, 2012.

[7]   C. J. Seema Maitreya, "MapReduce: Simplified Data Analysis of Big Data," *Procedia Computer Science,* vol. 57, pp. 563-571, 2015.

[8]   S. G. Jeffrey Dean, "MapReduce: Simplified Data Processing on Large Clusters," *USENIX Association OSDI,* vol. 4, pp. 137-149, 2004.

[9]   R. M. Yoo, A. Romano and C. Kozyrakis, "Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system," Austin, TX, USA, 2009.

[10] H. C. Y. D. C. B. M. Kyong-Ha Lee, "Parallel data processing with MapReduce: a survey," *ACM SIGMOD Record,* vol. 40, no. 4, 2012.

[11] B. P. J. S. H. S. B. R. J. Bayardo, "PLANET: Massively Parallel Learning of Tree Ensembles with MapReduce," *PVLDB,* vol. 2, no. 2, pp. 1426-1437, 2009.

[12] S. G. Jeffrey Dean, "MapReduce: simplified data processing on large clusters," *Communications of the ACM,* vol. 51, no. 1, 2008.

[13] J. Ekanayake, S. Pallickara and G. Fox, "MapReduce for Data Intensive Scientific Analyses," Indianapolis, IN, USA, 2008.

[14] A. Alam and J. Ahmed, "Hadoop Architecture and Its Issues," Las Vegas, NV, USA, 2014.

[15] R. K. R. R. Vijayakumari, "Comparative analysis of Google File System and Hadoop Distributed File System," *International Journal of Advanced Trends in Computer Science and Engineering,* vol. 3, no. 1, pp. 553-558, 2014.

[16] J. J. B. X. Y. F. Wang, "Hadoop high availability through metadata replication", in Proc," *The first international workshop on Cloud data management,* pp. 37-44, 2009.

[17] A. D. R.-L. H. D. S. P. Hung-chih Yang, "Map-reduce-merge: simplified relational data processing on large clusters," 2007.