Mobile Agent Based New Framework for Improving Big Data Analysis

Conference Paper · December 2013

DOI: 10.1109/CLOUDCOM-ASIA.2013.75

3 authors:

Youssef M. Essa
Bayer
12 PUBLICATIONS 65 CITATIONS
SEE PROFILE

Gamal Attiya
Faculty of Electronic Engineering
112 PUBLICATIONS 835 CITATIONS
SEE PROFILE

Ayman El-Sayed
Menoufia University
170 PUBLICATIONS 1,101 CITATIONS
SEE PROFILE

Mobile Agent based A New Framework for Improving Big Data Analysis

Youssef M. ESSA
Software Engineering Department,
Etisalat Corporation
Cairo, Egypt
yosuf_boray@yahoo.com

Gamal ATTIYA and Ayman EL-SAYED
Computer Science & Engineering Department
Faculty of Electronic Engineering, Menoufia Uni.
Menouf 32952, Egypt
gamal.attiya@yahoo.com
ayman.elsayed@el-eng.menofia.edu.eg

Abstract— **the rising number of applications serving millions of users and dealing with terabytes of data need to a faster processing paradigms. Recently, there is growing enthusiasm for the notion of big data analysis. Big data analysis becomes a very important aspect for growth productivity, reliability and quality of services (QoS). Processing of big data using a powerful machine is not efficient solution. So, companies focused on using Hadoop software for big data analysis. This is because Hadoop designed to support parallel and distributed data processing. However, Hadoop has several drawbacks effect on its performance and reliability against big data analysis. In this paper, a new framework is proposed to improve big data analysis and overcome the drawbacks of Hadoop. The proposed framework is called MapReduce Agent Mobility (MRAM). MRAM is developed by using mobile agent and MapReduce paradigm under Java Agent Development Framework (JADE).**

Index Terms — Mobile Agent; JADE; Big Data Analysis; HDFS; Fault Tolerance.

I. Introduction

The collection of data sets are so large and complex that become difficult to process using on-hand database management tools or traditional data processing applications referred to “Big Data”. The value of big data to an organization falls into two categories: analytical use and enabling new products. Extracting information and something intelligence from these big data sets, commonly referred to as big data analytics. Big data analytics can reveal insights hidden previously by data too costly to process such as peer influence among customers, revealed by analyzing shoppers’ transactions and social and geographical data. Big data analytics is shown to be useful in several scenarios; analytics enable web data mining and enable extracting business intelligence. The primary goal of big data analytics is to help companies to make better business decisions. But, analysis of large data sets in real-time requires a framework like MapReduce to distribute the work among tens, hundreds or even thousands of computers. So, many companies focused on using Hadoop for big data analysis.

Hadoop is an open source software framework written in Java by Doug cutting and Michael Cafarella [1]. Hadoop enables distributed, data intensive and parallel applications by dividing big data into smaller data blocks. These data blocks are divided into smaller partitions such that each data block processes a different partition in parallel [2]. By using Hadoop, there is no limit of storing and processing data by computational technique called MapReduce [3]. Hadoop provides a distributed file processing system that stores and processes a large scale of data [4]. It enables a fault tolerant by replicating data on three or more machines to avoid data loss [5], but this method causes some problems. The first problem is about increasing the amount of data that executes on machine by replicating each block of data in two or more machines, and second problem arises when the master machine is failed the full system is down.

So, in this paper presents a new strategy called MapReduce Agent Mobility (MRAM) to improve big data analysis and overcome the drawbacks of Hadoop. The proposed framework is developed by using mobile agent and MapReduce paradigm under Java Agent Development Framework (JADE). JADE is a promising middleware based on the agent paradigm because it supports generic services such as communication support, resource discovery, content delivery, data encoding and agents mobility [6, 7].

Indeed, there are seven reasons for using mobile agents as follows:

(1) Reduce the network load,
(2) Overcome network latency,
(3) Encapsulate protocols,
(4) Execute asynchronously and autonomously,
(5) Adapt dynamically,
(6) Naturally heterogeneous and robust, and
(7) Fault-tolerant [8].

So, the mobile agent is used with Hadoop to overcome the problems faced Hadoop. In the proposed strategy, mobile agents send both code and data to any machine. The machine can react dynamically for any changes in the environment. Furthermore, if a machine or environment down, the mobile agent can migrate to another machine with code and data.

The rest of this paper is organized as follows: Section II describes Hadoop architecture, workflow, and drawbacks. Section III presents the basic concepts of JADE and Mobile Agent. Section IV introduces the proposed framework namely MapReduce Agent Mobility (MRAM). Section V presents a comparative study and performance evaluation of the proposed strategy and Hadoop. Finally, the paper is concluded in Section VI.

II. HADOOP ARCHITECTURE AND WORKFLOW

This section presents both the architecture of Hadoop and its workflow for big data analysis as follow:

A. *Hadoop Architecture*

Hadoop architecture consists of a Hadoop Distributed File System (HDFS) and a programming framework MapReduce. HDFS stores big files across machines in a large cluster. Each file is stored as a sequence of blocks. Each block is sent to three or more machines for fault tolerance. Hadoop uses MapReduce method for processing data allocated on each node [9, 10, 11].

(1) *HDFS*

HDFS is a very large distributed file system [12, 13] that is available hardware and provides fault tolerance as well as have high throughput. Many big companies believe that within a few years, more than a half of the world's data will be stored in Hadoop. HDFS stores files as a series of blocks and replicates the data blocks for fault tolerance. HDFS is designed to store big data set, and provides global access to files in the cluster. HDFS stores metadata on a dedicated server, called "*NameNode*". Application data is stored on other servers called "*DataNodes*". All servers are fully connected and communicate with each other using TCP-based protocol [2, 12]. HDFS architecture is broadly divided into following four parts as the follows: *NameNode*, *DataNode*, *JobTracker* to determine the location of data and *TaskTracker* overseeing overall MapReduce job execution.

a) NameNode

The *NameNode* is responsible of managing all metadata and file system actions. It handles the file system namespace operations like open, close, and rename both file and directory Also, it makes all decisions regarding replication of blocks. *NameNode* maintains the tree of namespace and maps the file blocks to *DataNodes* (i.e. the physical location of file's data). A single *NameNode* is considered a bottleneck for handling requests in scientific application environments [9, 14].

b) DataNode

The *DataNode* stores data in the Hadoop file system, Each *DataNode* stores data blocks on behalf of local or remote clients. Each block is saved as a separated file in the node's local file system. On startup, *DataNode* connects to the *NameNode* and performs a handshake. The purpose of the handshake is to verify the namespace ID and the software version of *DataNode*. If *NameNode* does not match *DataNode*, the *DataNode* automatically shuts down. After the handshake is successful, the *DataNode* registers with the *NameNode*. *DataNode*s persistently store their unique storage IDs. The storage ID is an internal identifier of the *DataNode* which makes it as recognizable even if it is restarted with a different IP address or port. The storage ID is assigned to the *DataNode*, when it registers with the *NameNode* on the first time and never changes later. The *DataNode* then responds to the requests that coming from the *NameNode*, for the file system operations. The *DataNode*s service the read, writing and file replication requests based on the direction from which *NameNode* coming [5, 15].

c) JobTracker

The *JobTracker* talks to the *NameNode* to determine the location of the data. *JobTracker* schedules individual maps reduces or intermediate merging operations to specific machines. It monitors the success and failures of these individual tasks. Also, it works to complete the entire batch job. If a task fail, the *JobTracker* will automatically re-launch the task, possibly on a different node, up to a predefined limit of retries [14, 15].

d) TaskTracker

The *JobTracker* is the master overseeing the overall execution of a MapReduce job. The *TaskTracker*s manage the execution of individual tasks on each slave node. Although there is a single *TaskTracker* per slave node. Each *TaskTracker* can spawn multiple Java Virtual Machines (JVMs) to handle many maps or reduces the tasks in parallel. The *TaskTracker*s also transmit heartbeat messages to the *JobTracker*, usually every a few minutes, to reassure the *JobTracker* that is still a live [8, 14].

(2) *MapReduce*

In *MapReduce* [10], the first step is the map job which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job then takes the output from a map as input and combines those data tuples into a smaller set of pairs. The map function can run independently on each key/value pair, exposing enormous amounts of parallelism. Similarly, the reduce function can run independently on each intermediate key, exposing significant parallelism as well. Similar to other distributed systems, *MapReduce* also constitutes a master and a set of workers.The master is called *JobTracker*, while the workers are called *TaskTracker*s [11, 12].

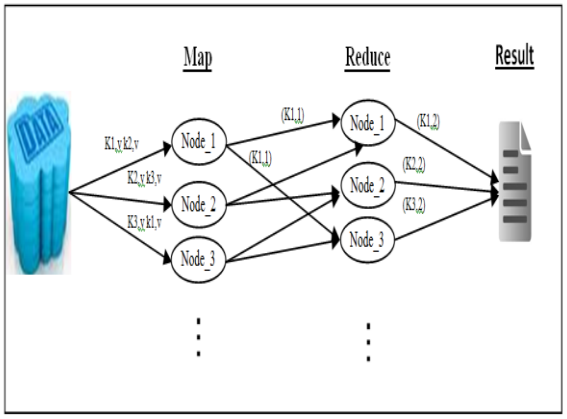

Figure 1. Workflow for Hadoop.

B. *Hadoop Workflow*

The workflow of Hadoop is shown in Figure 1. It has the following steps:

(1) Input text files to a platform.

(2) Server portioning file to blocks with the same size, then assigns a block of data to each computing node.

(3) The compute node runs map on the input data and producing intermediate data pair for every word, then sends its intermediate data pairs to the node designated to perform the reduce operation.

(4) The reduce operation counts the number of occurrences of each word using the values and emits it as a key-value pair.

(5) Server receives the results and outputs the list.

C. *Hadoop Drawbacks*

From the architecture of Hadoop and its workflow of data computation, there are many drawbacks of Hadoop. These drawbacks are:

(1) Hadoop Needs high memory and big storage to apply replication technique.

(2) Hadoop supports allocation of tasks only and do not have strategy to support scheduling of tasks.

(3) Still single master (*NameNode*) which requires care.

(4) Load time is long.

These drawbacks effects on both the performance and reliability of Hadoop against big data analysis. Therefore, it is necessary to develop a new framework or modify some Hadoop features to overcome Hadoop limitations and improve its performance and reliability. So, in this paper, a new framework is proposed to overcome the drawbacks of Hadoop and improve big data analysis.

III. Basic Concepts of JADE and Mobile Agent

A. *JADE Architectural Model*

In the recent years, there are many platforms that can support agent mobility and developing distributed application. JADE is a promising middleware based on the agent paradigm. It supports generic services such as communication support, resource discovery, content delivery, data encoding and so on [6, 7]. The architectural of JADE contains both the libraries required to develop application agents and the run-time environment that provides the basic services. These services include agent identification and agent communication. The instance of JADE is called "*Container*" and the set of all containers is called platform [7].

B. *Mobile Agent*

A mobile agent (MA) is a software abstraction that can migrate during execution across a heterogeneous or homogeneous network. It has the ability to suspend its execution according to some factors and resume it in another machine.

Characteristics of MA: There are several characteristics can be defined the structure of the MA [16]:

State: the main characteristic of the MA. It can stop execution on one machine and resume execution on another machine. The state depends on two factors:

1. Execution state, which is a runtime state including its program counter and stack.
2. Object state, which stores the current values of its variables.

Implementation: it is the program code that defines the tasks behaviour. If java is used as MA platform, classes present the implementation code. In this manner, there are two ways to make the required classes available to the MA:

1. Taking the entire required classes during its itinerary and uses it any time anywhere.
2. Taking some of the required classes and once the MA need a class that is not available, it retrieves it from remote location. This operation called *Code-On-Demand* technique, and it is a common technique in distributed network systems.

Interface: MA collaborates with other agents to handle the assign job. The Interface is required to make the communication possible between agents.

Unique Identifier: it is a unique ID define agent during its lifetime. It used as a key that needed to refer for a specific agent especially, when it travels all over the network.

Itinerary: it is the group of addresses created once the MA life start that define the agent journey around the network.

Principals: it is the information of individual, organization or corporation that MA belongs to. Principles are needed to authenticate the MA who dispatched to several destinations on the network.

Advantages of mobile agent: There are many advantages for using mobile agent to solve many problems on distributed application [6,7]:

Reduce network traffic: the cooperation in a distributed system is often achieved using communication protocols. These protocols transfer a large volumes of data stored at remote hosts over the network to a central processing site resulting in high network traffic. At this case, mobile agent uses alternative communication protocols.

Off-line tasks: network connections may be fail at any time. Agents can solved this problem by perform off-line tasks and send results to server application when it come back online.

Support for heterogeneous environments: MA can work on top of any operating system with the same its mobility framework.

Fault tolerance: Mobile agents react dynamically and autonomously to the changes in their environment. If a host is being shut down or platform is down, all agents executing on that machine will be warned and given time to dispatch themselves and continue their operation on another host in the network [17, 18].

Protocol Encapsulation: Protocol encapsulation allows the components of distributed system to communicate and coordinate their activities. MAs provide a solution to the problem of upgrading the protocol code at all locations in the distributed system.

IV. Our Proposed Framework

Our proposed framework is called *MapReduce Agent Mobility (MRAM)*. It combines the advantages of both mobile agent, and MapReduce technique.

A. MRAM Workflow

The workflow of the proposed MRAM framework is shown in Figure 2. It has the following steps:

(1) Input text files to the platform.

(2) Server portioning the file to blocks with the same size.

(3) Application server assigns a data block to each computing node, but in our approach the server take a task as the other nodes.

(4) The computing node runs map on the input data and producing intermediate data pair for every word. It then sends its intermediate data pairs to application server directly to perform the reduce operation.

(5) The reduce operation counts the number of occurrences of each word using the values and emits it as a key-value pair and save the result in file or in consol.

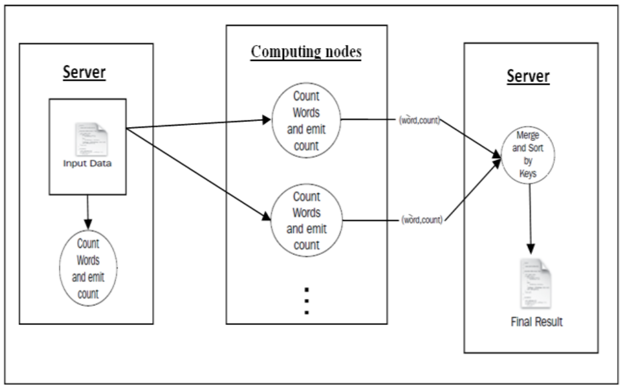

Figure 2. MRAM Workflow.

B. Advantages of MRAM

The MRAM framework has several advantages derived from the features of mobile agent and MapReduce technique. The advantages of MRAM are:

(1) Support allocation and scheduling tasks.

(2) Provides fault tolerance and don't need high memory or big disk to support it.

(3) Load time for MRAM is less than that of Hadoop.

(4) Solve single master (centralized node) problem by using features of mobile agent.

(5) Improve execution time because of no need to huge processing to replication data.

V. Comparative Study and Performance Analysis

It is noted that MRAM improves reliability of Hadoop using mobile agent and investigate the differences performance of Hadoop and MRAM. The idea of comparative study is applying the same application, the same environment on Hadoop and MRAM.

A. Implementation Environment

In this paper, the Measurements have been carried out by using the following hardware and software components. The specifications of the used hardware and software are shown in Table I.

(1) *Hardware Components:*
Hardware contains one server namely "Server" and three nodes namely PC1, PC2 and PC3 connected via a LAN.

(2) *Software Components*
Hadoop and MRAM are the main software components. The word count application is applying on each platform. It is a simple program given a text file and count repeated time for each word, after that save the output as a list in the form of (<Word>, <Count>). It is possible to process each line of a text file completely independently on the other lines. The data then is combined in a central location and the results are printed out.

TABLE I. Software and Hardware equipments

	Server	**PC1**	**PC2**	**PC3**
Model	IBM x3650	IBM		
CPU	Intel Dual Core2Quad 2.56 GHZ	Intel Dual Core2Due 2.53 GHz		
RAM	16 GB	2 GB		
Hadoop ver	0.20	0.20		
OS	Linux	Linux		
Sun JRE	JRE 7u25	JRE 7u25		
JADE	4.1	4.1		

B. Durability of platforms

As mentioned before, there are two weaknesses for Hadoop. The first, still single master which require care. The second, Hadoop reduce the CPU utilization by providing faults tolerance via replication data. Sure, these factors effect on the reliability of Hadoop, so the solution is via mobile agent as in MRAM and will clarify each solution separately as follow.

Centralized Node*:*

The Hadoop is still depend on single node that runs all the services needed to MapReduce task distribution and tracking. There is problem arising when single mode is fails or down causing the fall of the system. For this reason, we used features of mobile agent in our strategy that can react dynamically and autonomously to change in their environment. The idea of solution, the master node is selected when a platform starts working, that build linked list involve meta-data. This meta-data contains all information about the tasks, dependences among them and information about all machines. After that, the master node sends meta-data to all machines through the network. Subsequently, if any node receives a job, this node is elected as a new master node. If a master is shut down or platform is down, all agents executing on master machine will be moved including its code and data to another host that having a highest IP-address in the network. The agents continue their operation. This node becomes as a control node that responsible for all acts of server expect receive result from the machines such as *TaskTracker* and inform all machines about a machine failed. After all agents finished the executed tasks, it is waiting to send the result to general server when it come back online as shown in Figure 3.

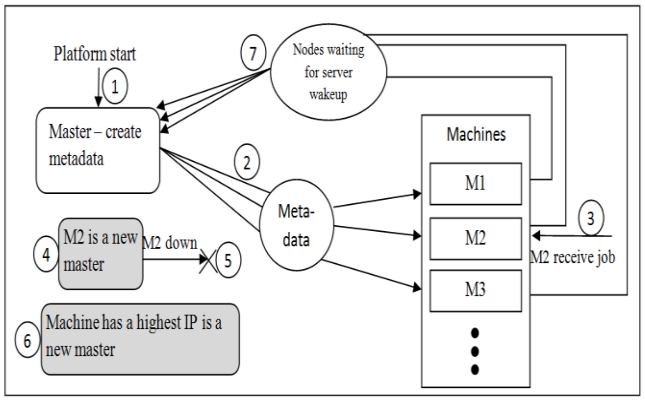

Figure 3. Solve centralized master node.

Faults Tolerance Techniques:

Hadoop reduces CPUs utilization by providing faults tolerance via replication data. Mobile agent provides fault tolerance by reacting dynamically and can move with its code and its data, to another machine to continue execute the task, if the framework is down or machine is shut down.

In Hadoop, each data block is sent to three or more node. New strategy is completely different, Figure 4 illustrates the steps to resolve this problem. The main idea is each data block is sent in mobile agent with code to specific machine, may be a framework down or machine shut down. In this case the agent react directly to change an environment and move to fastest machine available to continue execute tasks.

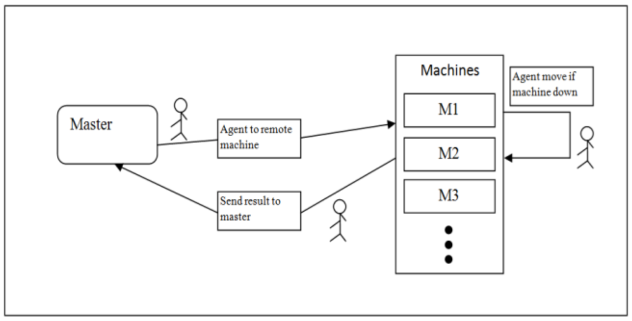

Figure 4. Flow Diagram if node down.

Performance Analysis:

The idea to evaluate the performance of two platforms via total time takes to complete assign job. The complete job is executed via both Hadoop and MRAM with different size of data. The execution time for each state is calculated. The load time and mapping task for Hadoop is larger than load time for MRAM because Hadoop takes time for replication processing. It means that each task is sent to three or more machines for fault tolerance, but in MRAM the task sent to only one machine and the mobility supports fault tolerance without needing for replication task.

In experimental study, Hadoop needs thirteen seconds to start to process a task but in MRAM need seven seconds only. Also, the two platforms using the same algorithm map reducing to evaluate execution time. The total time in MRAM is less than that of Hadoop as shown in Figure 5. MRAM gives the possibility for the server or control node to execute task as another nodes in platform, but not exist in Hadoop. Furthermore, Hadoop doesn't support scheduling tasks or can't work with dependent tasks but MRAM support this feature. Because of this reason, our proposal takes less execution time and high performance more than that of Hadoop.

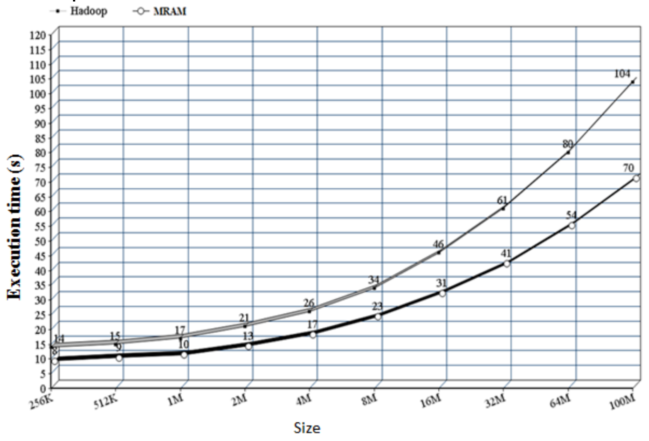

Figure 5. Comparison execution time between Hadoop and MRAM.

Summary of results:

Table II illustrates the differences between Hadoop and MRAM through the fetched results from experiments. The table shows various comparative factors.

TABLE II. SUMMARY ABOUT DIFFERENCES BETWEEN TWO PLATFORMS

Factors	Platforms	
	Hadoop	MRAM
Architecture	Client/Server	Distributed Agent
Startup time	Long	Less
Performance	Less	Better
Reliability	Reliable	More Reliable
Algorithm	Map-Reduce	Map-Reduce
Mobility	N/A	Support
Management disk	Support	N/A
Allocation Tasks	Support	Support
Scheduling Tasks	N/A	Support
Methodology	Object-Oriented	Object-Oriented
Language	Java	Java

VI. Conclusion

In this paper, a new framework called MRAM is developed using mobile agent and MapReduce paradigm under JADE. Our proposed framework is developed to improve big data analysis and to overcome the drawbacks of Hadoop. In the proposed Framework, mobile agents send both code and data to any machine and react dynamically for any changes in environment. In addition, the mobile agents have ability to move with code and data, if the machine or environment is down. Furthermore, Hadoop is still single master which requires care, this problem is solved in MRAM through send met-data contains map of network and all data about tasks and dependences between them. Also, MRAM improves performance by giving the server or control node, the possibility to execute tasks as the others nodes. Another disadvantage of Hadoop, it doesn't support scheduling tasks or does not work with dependent tasks, but MRAM support this feature. A new strategy is written in JAVA programming language based on JADE, This means it can run on different machines and different operating system without any problems.

References

[1] Hadoop web site, http://hadoop.apache.org/, Sep2013.

[2] Kala Karun. A, Chitharanjan. K, "A Review on Hadoop–HDFS Infrastructure Extensions", In Proceedings of IEEE Conference on Information and Communication Technologies (ICT2013), pp.132-137, 2013.

[3] Jian Tan, Xiaoqiao Meng, Li Zhang, "Coupling Task Progress for MapReduce Resource-Aware Scheduling", In Proceedings of IEEE INFOCOM, pp.1618-1626, 2013.

[4] S. Ghemawat, H. Gobioff, and S. Leung. "The google file system", In Proceedings of the nineteenth ACM symposium on Operating systems principles, SOSP '03, pp. 29–43, New York, NY, USA, 2003.

[5] K.Shvachko, H.Kuang, S.Radia, R.Chansler, "The Hadoop Distributed File System", 26th IEEE symposium on Mass Storage Systems and Technologies (MSST), pp.1-10, May, 2010.

[6] JADE web site, http://JADE.tilab.com, September, 2013.

[7] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, "JADE - A white paper" , TILAB Journal, Vol.3, No.3, pp.6-19, 2003.

[8] D.Schoder, T. Eymann, "Technical opinion: The real challenges of mobile agents", Communications of the ACM, Vol. 43, No. 6, pp. 111-112, June 2000.

[9] G.Mackey, S.Sehrish, J.Wang, "Improving Metadata Management for Small Files in HDFS", In Proceedings of IEEE International Conference on Cluster Computing and Workshops, pp.1-4, September, 2009.

[10] J. Dean, S. Ghemawat, "MapReduce: A Flexble Data Processing Tool", Communications of the ACM, Vol.53, No.1, pp.72-77, January, 2010.

[11] V.Martha, W.Zhao, Xiaowei Xu, "h-MapReduce: A Framework for Workload Balancing in MapReduce", IEEE 27th International Conference on Advanced Information Networking and Applications, pp.637-644, 25-28 March, 2013.

[12] J.Shafer, S.Rixner, Alan, "The Hadoop Distributed Filesystem:Balancing Portability and Performance", In Proceedings of IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS), pp.122–133, 28-30 March, 2010.

[13] K. Shvachko, H. Kuang, S. Radia, R. Chansler, "The Hadoop Distributed File System", 26th IEEE Symposium on Mass Storage Systems and technologies(MSST), pp. 1-10, 3-7 May, 2010.

[14] C.Lam, Hadoop in Action, Manning Publications, USA, 2011.

[15] S.Perera, T.Gunarathne, Hadoop MapReduce Cookbook, Packt Publishing, Feb 2013.

[16] D. Lange, M. Oshima, Programming and Deploying Java Mobile Agent with Aglets, Addison-Wesley, pp.18-20, 1998.

[17] P. Braun and W. Rossak, "Mobile Agents – Basic Concepts Mobility Models and the Tracy Toolkit", Morgan Kaufmann Publishers, 2005.

[18] L. Guanyu ; W. Baofeng; Y. Yang; A. Lihua , "Researches on Performance Optimization of Distributed Integrated System Based on Mobile Agent", The Sixth World Congress on Intelligent Control and Automation, pp. 4038- 4041, June 2006.

.