ISSN:0975-9646

Big Data Processing Using Hadoop MapReduce Programming Model

Anumol Johnson[#1]
Master Of Technology, Computer Science And Engineering
Sahrdaya college of Engineering And Technology
Calicut University, Kerala

Havinash P.H[#2]
Assistant Professor. Computer Science And Engineering
Sahrdaya college of Engineering And Technology
Calicut University, Kerala

Vince Paul[#3]
Head Of the Department, Computer Science And Engineering
Sahrdaya college of Engineering And Technology
Calicut University, Kerala

Mr.Sankaranarayanan P.N [#4]
Assistant Professor, Computer Science And Engineering
Sahrdaya college of Engineering And Technology
Calicut University, Kerala

***Abstract*— In today's age of information technology processing data is a very important issue. Nowadays even terabytes and petabytes of data is not sufficient for storing large chunks of database. The data is too big, moves too fast, or doesn't fit the structures of the current database architectures. Big Data is typically large volume of un-structured and structured data that gets created from various organized and unorganized applications, activities such as emails web logs, Facebook, etc. The main difficulties with Big Data include capture, storage, search, sharing, analysis, and visualization. Hence companies today use concept called Hadoop in their applications. Even sufficiently large amount of data warehouses are unable to satisfy the needs of data storage. Hadoop is designed to store large amount of data sets reliably. It is an open source software which supports parallel and distributed data processing. Along with reliability and scalability features Hadoop also provide fault tolerance mechanism by which system continues to function correctly even after some components fails working properly. Fault tolerance is mainly achieved using data duplication and making copies of same data sets in two or more data nodes. MapReduce is a programming model and an associated implementation for processing and generating large datasets that is flexible to a broad variety of real-world tasks. Users specify the computation in terms of a map and a reduce function, and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines.**

***Keywords*— Big data, Hadoop, Distributed file system, MapReduce**

I. INTRODUCTION

The emerging big-data paradigm, owing to its broader impact, has profoundly transformed our society and will continue to attract diverse attentions from both technological experts and the public in general. It is obvious that we are living a data deluge era, evidenced by the sheer volume of data from a variety of sources and its growing rate of generation. The exponential growth of data first presented challenges to cutting-edge businesses such as Google, Yahoo,
Amazon, Microsoft, Facebook, Twitter etc. Data volumes to be processed by cloud applications are growing much faster than computing power. For instance, an IDC report predicts that, from 2005 to 2020, the global data volume will grow by a factor of 300, from 130 exabytes to 40,000 exabytes, representing a double growth every two years. The term of "big-data" was coined to capture the profound meaning of this data-explosion trend and indeed the data has been touted as the new oil, which is expected to transform our society. The huge potential associated with big-data has led to an emerging research that has quickly attracted tremendous interest from diverse sectors, for example, industry, government and research community. Government has also played a major role in creating new programs to accelerate the progress of tackling the bigdata challenges. This growth demands new strategies for processing and analyzing information.

Hadoop has become a powerful Computation Model addresses to these problems. Hadoop HDFS became more popular amongst all the Big Data tools as it is open source with flexible scalability, less total cost of ownership and allows data stores of any form without the need to have data types or schemas defined. Hadoop MapReduce is a programming model and software framework for writing applications that rapidly process vast amounts of data in parallel on large clusters of compute nodes. Map reduce is a software frame work introduced by Google in 2004 to support distributed computing on large data sets on clusters of computers.The original MapReduce implementation by Google, as well as its open-source counterpart, Hadoop, is aimed for parallelizing computing in large clusters of

commodity machines. MapReduce model advantage is the easy scaling of data processing over multiple computing nodes.

II. BIG DATA

Big data is a collection of data sets so large and complex which is also exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn't fit the structures of our current database architectures. Big Data is typically large volume of un-structured (or semi structured) and structured data that gets created from various organized and computing problems across a number of servers. It is first developed and released as open source by Yahoo, it implements the MapReduce approach pioneered by Google in compiling its search indexes. Hadoops MapReduce involves distributing a dataset among multiple servers and operating on the data: the map stage. The partial results are then recombined: the reduce stage. To store data, Hadoop utilizes its own distributed file system, HDFS, which makes data available to multiple computing nodes.

Big data explosion, a result not only of increasing Internet usage by people around the world, but also the connection of billions of devices to the Internet. Eight years ago, for example, there were only around 5 exabytes of data online. Just two years ago, that amount of data passed over the Internet over the course of a single month. And recent estimates put monthly Internet data flow at around 21 exabytes of data. This explosion of data - in both its size and form - causes a multitude of challenges for both people and machines.

A. The Three V's of Big Data

Big data technologies describe a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data, by enabling high-velocity capture, discovery, and/or analysis. This definition delineates the three salient features of big data, i.e., volume, variety velocity. As a result, the 3Vs" definition has been used widely to characterize big data. First, the sheer volume of datasets is a critical factor for discriminating between big data and traditional data. For example, Facebook reports that its users registered 2.7 billion like" and comments per day in February 2012. Third, the velocity of big data means that datasets must be analyzed at a rate that matches the speed of data production. For time-sensitive applications, such as fraud detection and RFID data management, big data is injected into the enterprise in the form of a stream, which requires the system to process the data stream as quickly as possible to maximize its value. Finally, by exploiting a variety of mining methods to analyze big datasets, significant value can be derived from a huge volume of data with a low value density in the form of deep insight or commercial benefits.

1) Volume: We currently see the exponential growth in the data storage as the data is now more than text data. We can find data in the format of videos, musics and large images on our social media channels. It is very common to have Terabytes and Petabytes of the storage system for enterprises. As the database grows the applications and architecture built to support the data needsto be re-evaluated quite often. Sometimes the same data is re-evaluated with multiple angles and even though the original data is the same the new found intelligence creates explosion of the data. More sources of data are added on continuous basis. For companies, in the old days, all data was generated internally by employees. Currently, the data is generated by employees, partners and customers. For a group of companies, the data is also generated by machines. For example, Hundreds of millions of smart phones send a variety of information to the network infrastructure. This data did not exist five years ago. More sources of data with a larger size of data combine to increase the volume of data that has to be analyzed. This is a major issue for those looking to put that data to use instead of letting it just disappear. Petabyte data sets are common these days and Exabyte is not far away. The big volume indeed represents Big Data.

2) Velocity: Initially, companies analyzed data using a batch process. One takes a chunk of data, submits a job to the server and waits for delivery of the result. That scheme works when the incoming data rate is slower than the batch processing rate and when the result is useful despite the delay. With the new sources of data such as social and mobile applications, the batch process breaks down. The data is now streaming into the server in real time, in a continuous fashion and the result is only useful if the delay is very short. The data growth and social media explosion have changed how we look at the data. There was a time when we used to believe that data of yesterday is recent. The matter of the fact newspapers is still following that logic. However, news channels and radios have changed how fast we receive the news. Today, people reply on social media to update them with the latest happening. On social media sometimes a few seconds old messages (a tweet, status updates etc.) is not something interests users. They often discard old messages and pay attention to recent updates. The data movement is now almost real time and the update window has reduced to fractions of the seconds. This high velocity data represent Big Data.

3) Variety: Data can be stored in multiple formats. For example database, excel, csv, access or for the matter of the fact, it can be stored in a simple text le. Sometimes the data is not even in the traditional format as we assume, it may be in the form of video, SMS, pdf or something we might have not thought about it. One no longer has control over the input data format. Structure can no longer be imposed like in the past in order to keep control over the analysis. As new applications are introduced new data formats come to life. It is the need of the organization to arrange it and make it meaningful. It will be easy to do so if we have data in the same format, however it is not the case most of the time. The real world has data in many different formats and that is the challenge we need to overcome with the Big Data. This variety of the data represent Big Data.

III. INTRODUCTION TO HADOOP

Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library. Hadoop has its origins in Apache Nutch, an open source web search engine, itself a part of the Lucene project. Hadoop was the Doug Cuttings toy elephant .That's how the name came. Building a web search engine from scratch was an ambitious goal, for not only is the software required to crawl and index websites complex to write, but it is also a challenge to run without a dedicated operations team, since there are so many moving parts. It's expensive, too. MikeCafarella and Doug Cutting estimated a system supporting a 1-billion-page index would cost around half a million dollars in hardware, with a monthly running cost of dollar 30,000. Nevertheless, they believed it was a worthy goal, as it would open up and ultimately democratizesearch engine algorithms. In January 2008, Hadoop was made its own top-level project at Apache, confirming its success and its diverse, active community. By this time, Hadoop was being used by many other companies besides Yahoo!, such as Last.fm, Facebook, and the New York Times.

A. *Hadoop Related Projects*

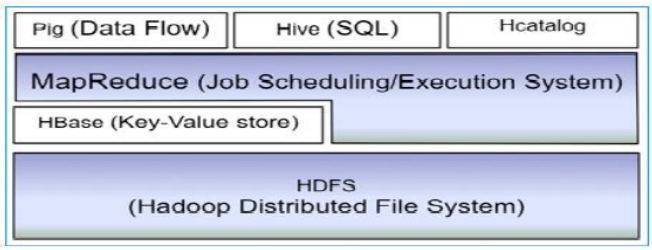

Fig 1: High Level Architecture of Hadoop

1) Hbase: HBase is a distributed, column-oriented database. HBase uses HDFS for its underlying storage. It maps HDFS data into a database like structure and provides Java API access to this DB. It supports batch style computations using MapReduce and point queries (random reads). HBase is used in Hadoop when random, realtime read/write access is needed. Its goal is the hosting of very large tables running on top of clusters of commodity hardware.

2) Pig: It is a data flow processing (scripting) language Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs. The main characteristic of Pig programs is that their structure can be substantially parallelized enabling them to handle very large data sets, simple syntax and advanced built-in functionality provide an abstraction that makes development of Hadoop jobs quicker and easier to write than traditional Java MapReduce jobs.

3) Zookeeper: It is a cluster configuration tool and distributed serialization manager useful to build large clusters of Hadoop nodes, high performance coordination service for distributed Cloud-based imageprocessing system with priority-based data distribution mechanism applications. ZooKeeper centralizes the services for maintaining the conguration information, naming, providing distributed synchronization, and providing group services.

4) Hive: Hive is a data warehouse infrastructure built on top of Hadoop. Hive provides tools to enable easy data summarization, ad-hoc querying and analysis of large datasets stored in Hadoop files. It provides a mechanism to put structure on this data and it also provides a simple query language called Hive QL, based on SQL, enabling users familiar with SQL to query this data.

5) Chukwa: It is used for monitoring large distributed clusters of servers. It is a data collection system for monitoring large distributed systems. Chukwa includes a flexible and powerful toolkit for displaying, monitoring and analyzing results to make the best use of the collected data.

6) HCatalog: It is a storage management layer for Hadoop that enables users with different data processing tools. HCatalog. s table abstraction presents users with a relational view of data in the Hadoop distributed file system (HDFS) and ensures that users need not worry about where or in what format their data is stored.

B. *Advantages*

Hadooop is highly scalable. The hadoop node is spread across the racks. The nodes inside one rack are connected by switch. The racks are connected by one or more core switches. Scalability indicates that ability to add node inside one cluster without any overhead. Since the nodes are spread across one rack, to add one node we need to consider only the rack configuration .No need of considering the entire clutster.So scaling on hadoop cluster is easy.Hadoop is economically cheap not only because of it is an openource platform but also the hadoop is build using cheap commodity of hardwares its cost is comparatively low compared to other data processing tool. Hadoop is flexible. Unlike parallel RDBMS we do not have to pre-process the data before using it. We donot have to design a star schema or update some data dictionary or manipulate it with a separate ETL process. Moreover we can change the schema after the fact with very little cost or e ort. There are some workloads where flexibility is very valuable and those workloads are moving to hadoop quickly.Also hadoop can process any type of data without considering it schema.Since the Namenode contionouly pings every Datanode, any fault occur to the Datanode can be easily found out.Each Datanode send report to its server in the form heartbeats.This include the information about the replicas and other block information. So the server can ensure that the worker is still alive and do their work. If the worker is not responding in a specified amount of time the worker is assumed to be dead.

IV. HADOOP DISTRIBUTED FILE SYSTEM

It is the primary storage system used by Hadoop applications. It is a distributed file system that provides high throughput access to application data creating multiple replicas of data blocks and distributing them on compute nodes throughout a cluster to enable reliable and rapid computations.

A. *Architecture*

HDFS architecture mainly composes of two components NameNode and DataNode. An HDFS cluster has two types of node operating in a master-worker pattern: a NameNode (the master) and a number of DataNodes (workers). The namenode manages the filesystem namespace.Datanodes are the workhorses of the filesystem. They store and retrieve blocks when they are told to (by clients or the namenode), and they report back to the namenode periodically with lists of blocks that they are storing.

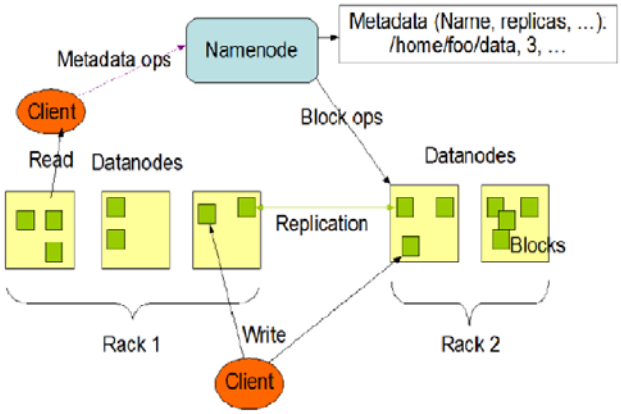

Fig 2: Hadoop Distributed Cluster File System Architechture

1) *Name Node:* Name Node decides about replication of data blocks. In a typical HDFS, block size is 64MB and replication factor is 3 (second copy on the local rack and third on the remote rack). The HDFS namespace is a hierarchy of files and directories. Files and directories are represented on the NameNode by inodes, which record attributes like permissions, modification and access times, namespace and disk space quotas. The file content is split into large blocks (typically128 megabytes, but user selectable file-by-le) and each block of the le is independently replicated at multiple DataNodes (typically three, but user selectable file-by-file). An HDFS client wanting to read a file first contacts the NameNode for the locations of data blocks comprising the file and then reads block contents from the DataNode closest to the client. When writing data, the client requests the NameNode to nominate a suite of three DataNodes to host the block replicas. The client then writes data to the DataNodes in a pipeline fashion.

HDFS keeps the entire namespace in RAM. The inode data and the list of blocks belonging to each file comprise the metadata of the name system called the image. The persistent record of the image stored in the local hosts native files system is called a checkpoint. The NameNode also stores the modification log of the image called the journal in the local hosts native file system. For improved durability, redundant copies of the checkpoint and journal can be made at other servers. During restarts the NameNode restores the namespace by reading the namespace and replaying the journal.

2) Data Node: Each block replica on a DataNode is represented by two files in the local hosts native file system.The first file contains the data itself and the second file is blocks metadata including checksums for the block data and the blocks generation stamp. During startup each 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 the DataNode. If either does not match that of the NameNode the DataNode automatically shuts down. After the handshake the DataNode registers with the NameNode. DataNodes persistently store their unique storage IDs. The storage ID is an internal identifer of the DataNode, which makes it recognizable even if it is restarted with a different IP address or port.

A DataNode identifies block replicas in its possession to the NameNode by sending a block report. A block report contains the block id, the generation stamp and the length for each block replica the server hosts. The first block report is sent immediately after the DataNode registration. Subsequent block reports are sent every hour and provide the NameNode with an up-todate view of where block replicas are located on the cluster. During normal operation DataNodes send heartbeats to the NameNode to con_rm that the DataNode is operating and the block replicas it hosts are available. The default heartbeat intervalis three seconds. If the NameNode does not receive a heartbeat from a DataNode in ten minutes the NameNode considers the DataNode to be out of service and the block replicas hosted by that DataNode to be unavailable.

3) HDFS Client: User applications access the _le system using the HDFS client, a code library that exports the HDFS file system interface. When an application reads a file, the HDFS client first asks the NameNode for the list of DataNodes that host replicas of the blocks of the file. It then contacts a DataNode directly and requests the transfer of the desired block. When a client writes, it first asks the NameNode to choose DataNodes to host replicas of the first block of the file. The client organizes a pipeline from node-to-node and sends the data. When the first block is filled, the client requests new DataNodes to be chosen to host replicas of the next block. A new pipeline is organized, and the client sends the further bytes of the file. Each choice of DataNodes is likely to be different.

B. *File I/O Operations*

Hadoop MapReduce applications use storage in a manner that is different from general-purpose computing. To read an HDFS file, client applications simply use a standard Java file input stream, as if the file was in the native filesystem. Behind the scenes, however, this stream is manipulated to retrieve data from HDFS instead. First, the Name Node is contacted to request access permission. If granted, the Name Node will translate the HDFS filename into a list of the HDFS block IDs comprising that file and a list of Data Nodes that store each block, and return the lists to the client. Next, the client opens a connection to the closest Data Node (based on Hadoop rack-awareness, but optimally the same node) and requests a specific block ID.

That HDFS block is returned over the same connection, and the data delivered to the application.

To write data to HDFS, client applications see the HDFS file as a standard output stream. Internally, however, stream data is first fragmented into HDFS-sized blocks (64MB) and then smaller packets (64kB) by the client thread. Each packet is enqueued into a FIFO that can hold up to 5MB of data, thus decoupling the application thread from storage system latency during normal operation. A second thread is responsible for dequeuing packets from the FIFO, coordinating with the Name Node to assign HDFS block IDs and destinations, and transmit-ting blocks to the Data Nodes (either local or remote) for storage. A third thread manages acknowledgements from the Data Nodes that data has been committed to disk.

V. MAP-REDUCE PROGRAMMINGG MODEL

MapReduce is a data processing or parallel programming model introduced by Google. In this model, a user specifies the computation by two functions, Map and Reduce. In the mapping phase, MapReduce takes the input data and feeds each data element to the mapper. In the reducing phase, the reducer processes all the outputs from the mapper and arrives at a final result. In simple terms, the mapper is meant to filter and transform the input into something that the reducer can aggregate over. The underlying MapReduce library automatically parallelizes the computation, and handles complicated issues like data distribution, load balancing and fault tolerance.

A. Architecture

The mapReduce is an master slave architecture as in HDFS.There are two types of nodes Task-tracker andJobtracker.Tasktracker act as master node and jobtracker as slave.The tasktracker divide the entire program into a number of individual program and give it to the workers.The worker compute each program individually and result are give back to Tasktracker. Job Tracker runs with the namenode ,receives the users job ,decides on how many tasks will run (number of mappers) and decides on where to run each mapper (concept of locality) Master pings workers periodically to detect failures.

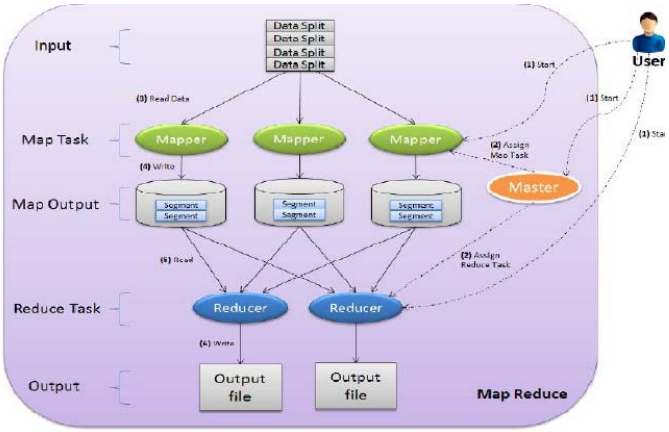

Fig 3: Architecture of MapReduce

MapReduce mainly composed of two phases Map and Reduce Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key is and passes them to the reduce function. The reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges these values together to form a possibly smaller set of values. Typically just zero or one output value is produced per reduce invocation. The intermediate values are supplied to the users to reduce function via an iterator.

This allows us to handle lists of values that are too large to fit in memory The Map function receives a key/value pair as input and generates intermediate key/value pairs to be further processed. The Reduce function merges all the intermediate key/value pairs associated with the same (intermediate) key and then generates final output. There are three main roles: the master, mappers, and reducers. The single master acts as the coordinator responsible for task scheduling, job management, etc. MapReduce is built upon a distributed file system (DFS) which provides distributed storage.

B. Execution Process in Mapreduce Programming Model

In MapReduce programming model and a MapReduce job consists of a map function, a reduce function, and When a function called the below steps of actions take place. MapReduce will first divide the data into N partitions with size varies from 16MB to 64MB. Then it will start many programs on a cluster of different machines. One of program is the master program; the others are workers, which can execute their work assigned by master.

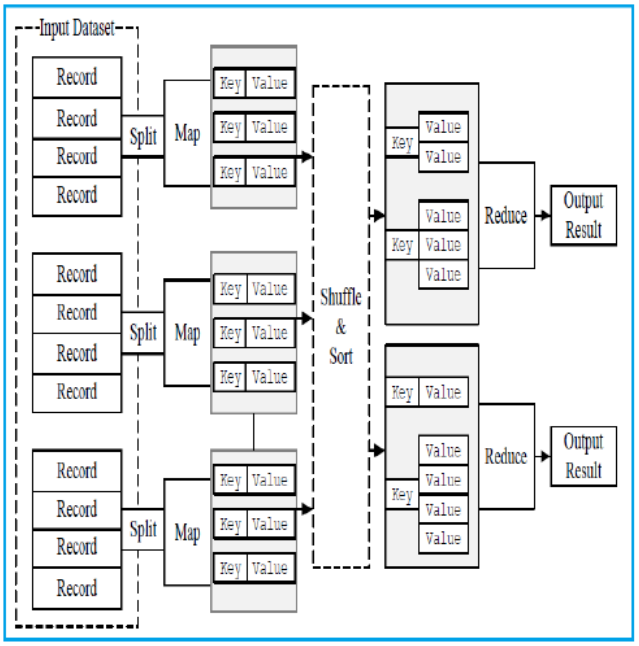

Fig 4: Execution process of MapReduce Programming Model

Master can distribute a map task or a reduce task to an idle worker. If a worker is assigned a Map task, it will parse the input data partition and output the key/value pairs, then pass

the pair to a user defined Map function. The map function will buffer the temporary key/value pairs in memory. The pairs will periodically be written to local disk and partitioned into P pieces. After that, the local machine will inform the master of the location of these pairs. If a worker is assigned a Reduce task and is informed about the location of these pairs, the Reducer will read the entire buffer by using remote procedure calls. After that, it will sort the temporary data based on the key. Then, the reducer will deal with all of the records. For each key and according set of values, the reducer passes key/value pairs to a user defined Reduce function.The output is the final output of this partition. After all of the mappers and reducers have finished their work, the master will return the result to users' programs. The output is stored in F individual files.

VI. Conclusion

The immense computational and storage power that a cloud infrastructure provides, can be aptly used to calculate large sets of data.Cloud technology progress and increased use of the Internet are creating very large new datasets with increasing value to businesses and processing power to analyze them affordable. BigData is still in its early infancy but it is already having a profound effect on technology companies and the way we do business. The size of these datasets suggests that exploitation may well require a new category of data storage and analysis systems with different architectures. Hadoop-MapReduce programming paradigm have a substantial base in the Big Data community due to the cost-effectiveness on commodity Linux clusters, and in the cloud via data upload to cloud vendors who have implemented Hadoop/HBase. The effectiveness and ease-of-use of the MapReduce method in parallelization involves many data analysis algorithms. HDFS, the Distributed File System, is a distributed file system designed to hold very large amounts of data (terabytes or even petabytes), and provide high-throughput access to these information.Despite some of the shortcomings like failures and breakdown of name node, Hadoop with HDFS provides a quite good enough way of handling faults tolerance.

Acknowledgment

I express my deepest thanks to “Mr. Havinah P H” the mentor of the seminar for guiding and correcting various documents of mine with attention and care. He has taken the pain to go through the seminar and make necessary correction as and when needed. I also extended my heartfelt thanks to my family and well wishers.

References

[1] R. Taylor. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics BMC bioinformatics,11(Suppl 12):S1, 2010.

[2] A. Pavlo et al . A comparison of approaches to large-scale data analysis. In Proceedings of the ACM SIGMOD, pages 165178, 2009.

[3] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (2009) 599616.

[4] Hadoop Distributed File Systemhttp://hadoop.apache.org/hdfs[3] Borthakur, D. (2007) The Hadoop Distributed File System: Architecture and Design.http://hadoop.apache.org/common/docs/r0.18.0/hdfs_design.pdf

[5] W. Jiang et al . A Map-Reduce System with an Alternate API for Multi-core Environments. In Proceedings of the 10th IEEE/ACM CCGrid, pages 8493, 2010.

[6] Map-Reduce: Simplied Data Processing on LargeClusters, by Jerey Dean and SanjayGhemawat; fromGoogle Research.