Concepts and Technologies of Big Data Management and Hadoop File System

Balu Srinivasulu[#1], Andemariam Mebrahtu [#2]

[1, 2] *Lecturer, Computer Science, Eritrea Institute of Technology*
Asmara, Eritrea, North East Africa

Abstract: *Big data is a broad term for data sets so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, data curation, search, sharing, storage, transfer, visualization, and information privacy.The term often refers simply to the use of predictive analytics or other certain advanced methods to extract value from data, and seldom to a particular size of data set. Accuracy in big data may lead to more confident decision making. And better decisions can mean greater operational efficiency, cost reductions and reduced risk.Analysis of data sets can find new correlations, to "spot business trends, prevent diseases, combat crime and soon." Scientists, practitioners of media and advertising and governments alike regularly meet difficulties with large data sets in areas including Internet search, finance and business informatics. Scientists encounter limitations in e-Science work, including meteorology, genomics, connectionism, complex physics simulations, and biological and environmental research.Work with big data is necessarily uncommon; most analysis is of "PC size" data, on a desktop PC or notebook that can handle the available data set. Relational database management systems and desktop statistics and visualization packages often have difficulty handling big data. The work instead requires "massively parallel software running on tens, hundreds, or even thousands of servers". What is considered "big data" varies depending on the capabilities of the users and their tools, and expanding capabilities make Big Data a moving target. This paper presents an overview of big data technologies Map Reduce and Hadoop environment the current issues with these technologies.*

Keywords: *Big Data, Hadoop, Map Reduce*

I. INTRODUCTION

Big data is a blanket term for the non-traditional strategies and technologies needed to gather, organize, process, and gather insights from large datasets. While the problem of working with data that exceeds the computing power or storage of a single computer is not new, the pervasiveness, scale, and value of this type of computing has greatly expanded in recent years**.**

Big Data Innovations in technology and greater affordability of digital devices have presided over today's Age of Big Data, an umbrella term for the explosion in the quantity and diversity of high frequency digital data. These data hold the potential as yet largely untapped to allow decision makers to track development progress, improve social protection, and understand where existing policies and programmers require adjustment [2].

Turning Big Data call logs, mobile-banking transactions, online user-generated content such as blog posts and Tweets, online searches, satellite images, etc. into actionable information requires using computational techniques to unveil trends and patterns within and between these extremely large socioeconomic datasets. New insights gleaned from such data mining should complement official statistics, survey data, and information generated by Early Warning Systems, adding depth and nuances on human behaviors and experiences and doing so in real time, thereby narrowing both information and time gaps. With the promise come questions about the analytical value and thus policy relevance of this data including concerns over the relevance of the data in developing country contexts, its representativeness, its reliability as well as the overarching privacy issues of utilizing personal data. This paper does not offer a grand theory of technology-driven social change in the Big Data era [1]. Rather it aims to delineate the main concerns and challenges raised by "Big Data for Development" as concretely and openly as possible, and to suggest ways to address at least a few aspects of each. Big Data is a collection of large datasets that cannot be processed using traditional computing techniques. It is not a single technique or a tool; rather it involves many areas of business and technology.

Fig.1 Big Data Diagram

II. IMPORTANCE OF BIG DATA

The government's emphasis is on how big data creates "value" – both within and across disciplines and domains. Value arises from the ability to analyze the data to develop actionable information. The survey of the technical literature [3] suggests five generic ways that big data can support value creation for organizations.

a) Creating transparency by making big data openly available for business and functional analysis (quality, lower costs, reduce time to market, etc.).
b) Supporting experimental analysis in individual locations that can test decisions or approaches, such as specific market programs.
c) Assisting, based on customer information, in defining market segmentation at more narrow levels.
d) Supporting Real-time analysis and decisions based on sophisticated analytics applied to data sets from customers and embedded sensors.
e) Facilitating computer-assisted innovation in products based on embedded product sensors indicating customer responses.

III. BI G DATA TYPES:

A. Structured Data

Structured Data are numbers and words that can be easily categorized and analyzed. These data are generated by things like network sensors embedded in electronic devices, smart phones, and global positioning system (GPS) devices. Structured data also include things like sales figures, account balances, and transaction data.

B. Unstructured Data

Unstructured Data include more complex information, such as customer reviews from commercial websites, photos and other multimedia, and comments on social networking sites. These data cannot easily be separated into categories or analyzed numerically.

The explosive growth of the Internet in recent years means that the variety and amount of big data continue to grow. Much of that growth comes from unstructured data.

C. Semi-structured data

Semi-structured data is a form of structured data that does not conform to the formal structure of data models associated with relational databases or other forms of data tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data. Example: XML data.

IV. BIG DATA CHARACTERISTICS

Big Data as having three dimensions: volume, variety, and velocity. Thus, IDC defined it: 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." [4] Two other characteristics seem relevant: value and complexity. We summarize these characteristics as given below.

A. Data Volume

Data volume measures the amount of data available to an organization, which does not necessarily have to own all of it as long as it can access it. As data volume increases, the value of different data records will decrease in proportion to age, type, richness, and quantity among other factors.

B. Data Velocity

Data velocity measures the speed of data creation, streaming, and aggregation. Ecommerce has rapidly increased the speed and richness of data used for different business transactions (for example, web-site clicks). Data Variety: Data variety is a measure of the richness of the data representation – text, images video, audio, etc.

C. Data Variety

Data variety measures the usefulness of data in making decisions. It has been noted that "the purpose of computing is insight, not numbers". Data science is exploratory and useful in getting to know the data, but "analytic science" encompasses the predictive power of big data.

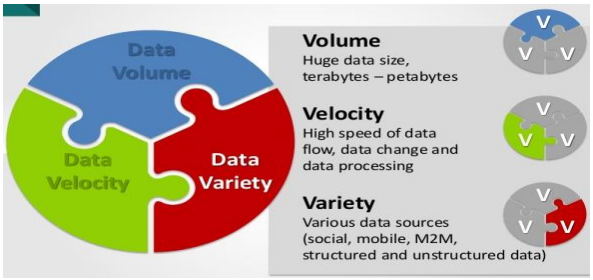

Fig. 2 Big Data Characteristics diagram

V. WHAT COMES UNDER BIG DATA

Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

- Black Box Data: It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.
- Social Media Data: Social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.
- Stock Exchange Data: The stock exchange data holds information about the 'buy' and 'sell' decisions made on a share of different companies made by the customers.
- Power Grid Data: The power grid data holds information consumed by a particular node with respect to a base station.
- Transport Data: Transport data includes model, capacity, distance and availability of a vehicle.
- Search Engine Data: Search engines retrieve lots of data from different databases.

A. Operational Big Data

These include systems like MongoDB that provide operational capabilities for real-time, interactive workloads where data is primarily captured and stored.

NoSQL Big Data systems are designed to take advantage of new cloud computing architectures that have emerged over the past decade to allow massive computations to be run inexpensively and efficiently. This makes operational big data workloads much easier to manage, cheaper, and faster to implement.

Some NoSQL systems can provide insights into patterns and trends based on real-time data with minimal coding and without the need for data scientists and additional infrastructure.

B. Analytical Big Data

These includes systems like Massively Parallel Processing (MPP) database systems and MapReduce that provide analytical capabilities for retrospective and complex analysis that may touch most or all of the data.

Map Reduce provides a new method of analyzing data that is complementary to the capabilities provided by SQL, and a system based on MapReduce that can be scaled up from single servers to thousands of high and low end machines.

These two classes of technology are complementary and frequently deployed together.

	Operational	**Analytical**
Latency	1 ms - 100 ms	1 min - 100 min
Concurrency	1000 - 100,000	1 – 10
Access Pattern	Writes and Reads	Reads
Queries	Selective	Unselective
Data Scope	Operational	Retrospective
End User	Customer	Data Scientist
Technology	NoSQL MapReduce	MPP Database

VI. BIG DATA LIFE CYCLE PROCESS

So how is data actually processed when dealing with a big data system? While approaches to implementation differ, there are some commonalities in the strategies and software that we can talk about generally. While the steps presented below might not be true in all cases, they are widely used.

The general categories of activities involved with big data processing are:

- Ingesting data into the system
- Persisting the data in storage
- Computing and Analyzing data
- Visualizing the results

Before we look at these four workflow categories in detail, we will take a moment to talk about **clustered computing**, an important strategy employed by most big data solutions. Setting up a computing cluster is often the foundation for technology used in each of the life cycle stages.

A. Clustered Computing

Because of the qualities of big data, individual computers are often inadequate for handling the data at most stages. To better address the high storage and computational needs of big data, computer clusters are a better fit.

Big data clustering software combines the resources of many smaller machines, seeking to provide a number of benefits:

Resource Pooling: Combining the available storage space to hold data is a clear benefit, but CPU and memory pooling is also extremely important.

Processing large datasets requires large amounts of all three of these resources.

- **High Availability**: Clusters can provide varying levels of fault tolerance and availability guarantees to prevent hardware or software failures from affecting access to data and processing. This becomes increasingly important as we continue to emphasize the importance of real-time analytics.
- **Easy Scalability**: Clusters make it easy to scale horizontally by adding additional machines to the group. This means the system can react to changes in resource requirements without expanding the physical resources on a machine.

Using clusters requires a solution for managing cluster membership, coordinating resource sharing, and scheduling actual work on individual nodes. Cluster membership and resource allocation can be handled by software like **Hadoop's YARN** (which stands for yet another Resource Negotiator) or **Apache Mesos**.

The assembled computing cluster often acts as a foundation which other software interfaces with to process the data. The machines involved in the computing cluster are also typically involved with the management of a distributed storage system, which we will talk about when we discuss data persistence.

B. Ingesting Data into the System

Data ingestion is the process of taking raw data and adding it to the system. The complexity of this operation depends heavily on the format and quality of the data sources and how far the data is from the desired state prior to processing.

One way that data can be added to a big data system is dedicated ingestion tools. Technologies like **Apache Sqoop** can take existing data from relational databases and add it to a big data system. Similarly, **Apache Flume** and **Apache Chukwa** are projects designed to aggregate and import application and server logs. Queuing systems like **Apache Kafka** can also be used as an interface between various data generators and a big data system. Ingestion frameworks like **Gobblin** can help to aggregate and normalize the output of these tools at the end of the ingestion pipeline.

During the ingestion process, some level of analysis, sorting, and labeling usually takes place. This process is sometimes called ETL, which stands for extract, transform, and load. While this term conventionally refers to legacy data warehousing processes, some of the same concepts apply to data entering the big data system. Typical operations might include modifying the incoming data to format it, categorizing and labeling data, filtering out unneeded or bad data, or potentially validating that it adheres to certain requirements.

With those capabilities in mind, ideally, the captured data should be kept as raw as possible for greater flexibility further on down the pipeline.

C. Persisting the Data in Storage

The ingestion processes typically hand the data off to the components that manage storage, so that it can be reliably persisted to disk. While this seems like it would be a simple operation, the volume of incoming data, the requirements for availability, and the distributed computing layer make more complex storage systems necessary.

This usually means leveraging a distributed file system for raw data storage. Solutions like **Apache Hadoop's HDFS** file system allow large quantities of data to be written across multiple nodes in the cluster. This ensures that the data can be accessed by compute resources, can be loaded into the cluster's RAM for in-memory operations, and can gracefully handle component failures. Other distributed file systems can be used in place of HDFS including **Ceph**and **GlusterFS**.

Data can also be imported into other distributed systems for more structured access. Distributed databases, especially NoSQL databases, are well-suited for this role because they are often designed with the same fault tolerant considerations and can handle heterogeneous data. There are many different types of distributed databases to choose from depending on how you want to organize and present the data.

D. Computing and Analyzing Data

Once the data is available, the system can begin processing the data to surface actual information. The computation layer is perhaps the most diverse part of the system as the requirements and best approach can vary significantly depending on what type of insights desired. Data is often processed repeatedly, either

iteratively by a single tool or by using a number of tools to surface different types of insights.

Batch processing: is one method of computing over a large dataset. The process involves breaking work up into smaller pieces, scheduling each piece on an individual machine, reshuffling the data based on the intermediate results, and then calculating and assembling the final result. These steps are often referred to individually as splitting, mapping, shuffling, reducing, and assembling, or collectively as a distributed map reduce algorithm. This is the strategy used by **Apache Hadoop's MapReduce**. Batch processing is most useful when dealing with very large datasets that require quite a bit of computation.

While batch processing is a good fit for certain types of data and computation, other workloads require more real-time processing. Real-time processing demands that information be processed and made ready immediately and requires the system to react as new information becomes available. One way of achieving this is stream processing, which operates on a continuous stream of data composed of individual items. Another common characteristic of real-time processors is in-memory computing, which works with representations of the data in the cluster's memory to avoid having to write back to disk.

Apache Storm, **Apache Flink**, and **Apache Spark** provide different ways of achieving real-time or near real-time processing. There are trade-offs with each of these technologies, which can affect which approach is best for any individual problem. In general, real-time processing is best suited for analyzing smaller chunks of data that are changing or being added to the system rapidly.

The above examples represent computational frameworks. However, there are many other ways of computing over or analyzing data within a big data system. These tools frequently plug into the above frameworks and provide additional interfaces for interacting with the underlying layers. For instance,

Apache Hive: provides a data warehouse interface for Hadoop, **Apache Pig** provides a high level querying interface, while SQL-like interactions with data can be achieved with projects like **Apache Drill**, **Apache Impala**, **Apache Spark SQL**, and **Presto**. For machine learning, projects like **Apache SystemML**, **Apache Mahout**, and **Apache Spark's MLlib** can be useful. For straight analytics programming that has wide support in the big data ecosystem, both **R** and **Python** are popular choices.

E. Visualizing the Results

Due to the type of information being processed in big data systems, recognizing trends or changes in data over time is often more important than the values themselves. Visualizing data is one of the most useful ways to spot trends and make sense of a large number of data points.

Real-time processing is frequently used to visualize application and server metrics. The data changes frequently and large deltas in the metrics typically indicate significant impacts on the health of the systems or organization. In these cases, projects like **Prometheus** can be useful for processing the data streams as a time-series database and visualizing that information.

One popular way of visualizing data is with the **Elastic Stack**, formerly known as the ELK stack. Composed of Logstash for data collection, Elastic search for indexing data and Kibana for visualization, the Elastic stack can be used with big data systems to visually interface with the results of calculations or raw metrics. A similar stack can be achieved using **Apache Solr** for indexing and a Kibana fork called **Banana** for visualization. The stack created by these is called **Silk**.

Another visualization technology typically used for interactive data science work is a data "notebook". These projects allow for interactive exploration and visualization of the data in a format conducive to sharing, presenting, or collaborating. Popular examples of this type of visualization interface are **Jupyter Notebook** and **Apache Zeppelin**.

VII. HADOOP

Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel with others. In short, Hadoop is used to develop applications that could perform complete statistical analysis on huge amounts of data.

This is a freely available java based programming framework supporting for the processing of large sets of data in a distributed computing environment. Using Hadoop, big amount of data sets can be processed over cluster of servers and apps may be run on system with thousands of nodes involving terabytes of information. This lowers the risk of

system failure even when a huge number of nodes fail. it enables a scalable, flexible, fault tolerant computing solution. HDFS [2], a file system spanning all nodes in a Hadoop cluster for data storage links the file systems on local nodes to make it onto a very large file system thus improving the reliability.

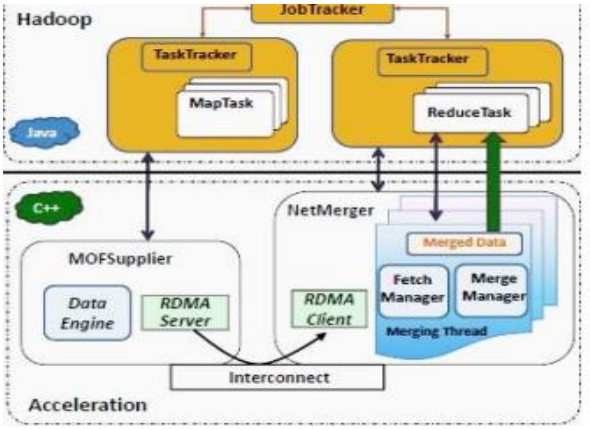

Fig. 3 Hadoop Structure

- Task trackers are responsible for running the tasks that the job tracker assigns them
- Job trackers has two primary responsibilities which are managing the cluster resources and scheduling all user jobs
- Data engine consists of all the information about the processing the data
- Fetch manager helps to fetch the data while particular task is running.

VIII. MAP REDUCE

Map reduces [5] framework is used to write apps that process large amounts of data in a reliable and fault tolerant way. The application is initially divided into individual chunks which are processed by individual map jobs in parallel. The output of map sorted by a framework and then sent to the reduce tasks. The monitoring is taken care by the framework

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework

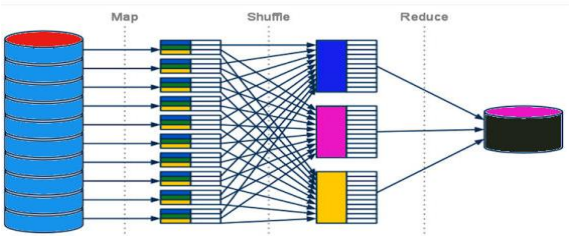

Fig. 4 Map Reduce diagram

IX. HADOOP DISTRIBUTED FILE SYSTEM

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets [6].

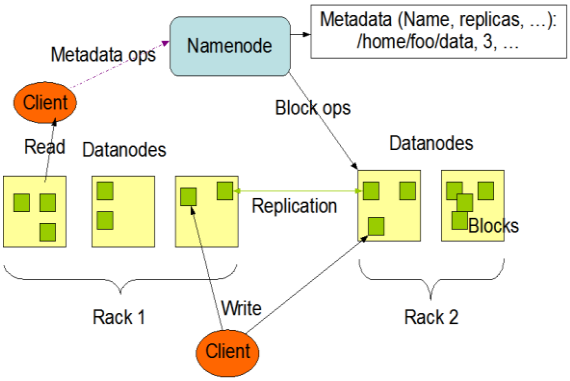

Fig. 5 HDFS Structure

HDFS has master/slave architecture. An HDFS cluster consists of a single Name Node, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of Data Nodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of Data Nodes.

The Name Node executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to Data Nodes. The Data Nodes are responsible for serving read and write requests from the file system's clients. The Data Nodes also perform block creation, deletion, and replication upon instruction from the Name Node.

The Name Node and Data Node are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the Name Node or the Data Node software. Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the Name Node software. Each of the other machines in the cluster runs one instance of the Data Node software. The architecture does not preclude running multiple Data Nodes on the same machine but in a real deployment that is rarely the case.
The existence of a single Name Node in a cluster greatly simplifies the architecture of the system. The Name Node is the arbitrator and repository for all HDFS metadata. The system is designed in such a way that user data never flows through the Name Node.

A. The File System Namespace

HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories. The file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from one directory to another, or rename a file. HDFS does not yet implement user quotas or access permissions. HDFS does not support hard links or soft links. However, the HDFS architecture does not preclude implementing these features.

The Name Node maintains the file system namespace. Any change to the file system namespace or its properties is recorded by the Name Node. An application can specify the number of replicas of a file that should be maintained by HDFS. The number of copies of a file is called the replication factor of that file. This information is stored by the Name Node.

B. Data Replication

HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are writing-once and have strictly one writer at any time. The Name Node makes all decisions regarding replication of blocks. It periodically receives a Heartbeat and a Block report from each of the Data Nodes in the cluster. Receipt of a Heartbeat implies that the Data Node is functioning properly. A Block report contains a list of all blocks on a Data Node.

X. ASSUMPTIONS AND GOALS OF HDFS

- ***Hardware Failure***

Hardware failure is the norm rather than the exception. An HDFS instance may consist of hundreds or thousands of server machines, each storing part of the file system's data. The fact that there are a huge number of components and that each component has a non-trivial probability of failure means that some component of HDFS is always non-functional. Therefore, detection of faults and quick, automatic recovery from them is a core architectural goal of HDFS.

- ***Streaming Data Access***

Applications that run on HDFS need streaming access to their data sets. They are not general purpose applications that typically run on general purpose file systems. HDFS is designed more for batch processing rather than interactive use by users. The emphasis is on high throughput of data access rather than low latency of data access. POSIX imposes many hard requirements that are not needed for applications that are targeted for HDFS. POSIX semantics in a few key areas has been traded to increase data throughput rates.

- ***Large Data Sets***

Applications that run on HDFS have large data sets. A typical file in HDFS is gigabytes to terabytes in size. Thus, HDFS is tuned to support large files. It should provide high aggregate data bandwidth and scale to hundreds of nodes in a single cluster. It should support tens of millions of files in a single instance.

- ***Simple Coherency Model***

HDFS applications need a write-once-read-many access model for files. A file once created, written, and closed need not be changed except for appends and truncates. Appending the content to the end of

the files is supported but cannot be updated at arbitrary point. This assumption simplifies data coherency issues and enables high throughput data access. A MapReduce application or a web crawler application fits perfectly with this model.

- ***Moving Computation is Cheaper than Moving Data***

A computation requested by an application is much more efficient if it is executed near the data it operates on. This is especially true when the size of the data set is huge. This minimizes network congestion and increases the overall throughput of the system. The assumption that it is often better to migrate the computation closer to where the data is located rather than moving the data to where the application is running. HDFS provides interfaces for applications to move themselves closer to where the data is located.

- ***Portability across Heterogeneous Hardware and Software Platforms***

HDFS has been designed to be easily portable from one platform to another. This facilitates widespread adoption of HDFS as a platform of choice for a large set of applications.

XI. CONCLUSION

This paper gave a description of a systematic flow of survey of the big data and Big Data will simply replace the approaches, tools and systems that underpin development work. What it does say, however, is that Big Data constitutes an historic opportunity to advance our common ability to support and protect human communities by understanding the information they increasingly produce in digital forms. Big data is a broad, rapidly evolving topic. While it is not well-suited for all types of computing, many organizations are turning to big data for certain types of work loads and using it to supplement their existing analysis and business tools. Big data systems are uniquely suited for surfacing difficult-to-detect patterns and providing insight into behaviors that are impossible to find through conventional means. By correctly implement systems that deal with big data, organizations can gain incredible value from data that is already available.

REFERENCES

[1]. DunrenChe, MejdlSafran, ZhiyongPeng,"From Big Data to Big Data Mining: Challenges, Issues, and Opportunities", DASFAA Workshops 2013, LNCS 7827, pp. 1–15, 2013

[2]. Venkata Narasimha Inukollu , Sailaja Arsi and Srinivasa Rao Ravuri "Security issues associated with big data in cloud computing "International Journal of Network Security & Its Applications (IJNSA), Vol.6, No.3, May 2014.

[3]. Dai, Jinquan, et al.,"Hitune: dataflow-based performance analysis for big data cloud", Proc. of the 2011 USENIX ATC (2011), pp. 87-100. [Online]Available:https://www.usenix.org/legacy/event/atc 11/tech/final_files/Dai.pdf.

[4]. K, Chitharanjan, and Kala Karun A. "A review on hadoop — HDFS infrastructure extensions." JeJu Island: 2013, pp. 132-137, 11-12 Apr. 2013.

[5]. Lohr, Steve. "The Age of Big Data." New York Times. 11 Feb, 2012. http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html?_r=2&pagewanted=all

[6]. D. Borthakur, "The hadoop distributed fi le system: Architecture and design," Hadoop Project Website, vol. 11, 2007

[7]. Wie, Jiang, Ravi V.T, and Agrawal G. "A Map-Reduce System with an Alternate API for Multi-core Environments." Melbourne, VIC: 2010, pp. 84-93, 17-20 May. 2010

[8]. Jefry Dean, Sanjay Ghemwat,"Mapreduce: A Flexible Data Processing Tool", communications of the ACM, Vol. 53, Issuse 1, January 2010, pp. 72-77.

[9]. Manyika, James, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh and Angela H. Byers. "Big data: The next frontier for innovation, competition, and productivity." McKinsey Global Institute (2011): 1-137. May 2011.

[10]. ! Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute, June 2011.http://www.mckinsey.com/mgi/publications/big_data/pdfs/MGI_big_data_full_report.pdf

[11]. Boyd, Dana and Crawford, Kate. "Six Provocations for Big Data." Working Paper - Oxford Internet Institute. 21 Sept. 2011http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1926431>

[12]. Villars, R. L., Olofson, C. W., & Eastwood, M. (2011, June). Big data: What it is and why you should care. IDC White Paper. Framingham, MA: IDC.

[13]. F.C.P, Muhtaroglu, Demir S, Obali M, and Girgin C. "Busines on big dataapplications." Big Data, 2013 IEEE International Conference, Silicon Valley, CA, Oct 6-9, 2013, pp.32 - 37.

BIOGRAPHIES

Mr. Balu Srinivasulu currently I am working as a Lecturer in the Department of Computer Science, Eritrea Institute of Technology, Asmara, Eritrea. I have wide experience of teaching and research in field of Computer Science. I have published a number of international journal papers related to the Computer Science. My areas of research are Wireless Networks, Communication Networks, Big Data and Cloud Computing.

Mr. Andemariam Mebrahtu, Currently I am working as Lecturer and HOD in Department of Computer Science, Eritrea Institute of Technology, Asmara, Eritrea. I have sound experience in teaching and academic Administration activities. My area of interest includes Cloud Computing, Distributed Computing and Big Data Management.