International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 4 Issue 1, December 2019 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

A Review Paper on Big Data and Hadoop for Data Science

Mr. Ketan Bagade[1], Mrs. Anjali Gharat[2], Mrs. Helina Tandel[3]

[1]Faculty of Information Technology, [2]Faculty of Computer Engineering,
[3]Faculty of Electronics & Telecommunications,
[1,2,3]Vidyalankar Polytechnic, Wadala, Mumbai, Maharashtra, India

ABSTRACT

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 has become a complete subject, which involves various tools, technqiues and frameworks. Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

KEYWORDS: *Big Data, Hadoop, Map Reduce, HDFS, Hadoop Components*

How to cite this paper: Mr. Ketan Bagade | Mrs. Anjali Gharat | Mrs. Helina Tandel "A Review Paper on Big Data and Hadoop for Data Science" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-4 | Issue-1, December 2019, pp.1216-1221, URL: www.ijtsrd.com/papers/ijtsrd29816.pdf

IJTSRD29816

1. INRODUCTION

A. What is BigData

Big Data is also **data** but with a **huge size**. Big Data is a term used to describe a collection of data that is huge in size and yet growing exponentially with time. In short such data is so large and complex that none of the traditional data management tools are able to store it or process it efficiently.

B. 3 Vs of Big Data

Volume of data: Volume refers to amount of data. Volume of data stored in enterprise repositories have grown from megabytes and gigabytes to petabytes.

Variety of data: Different types of data and sources of data. Data variety exploded from structured and legacy data stored in enterprise repositories to unstructured, semi structured, audio, video, XML etc.

Velocity of data: Velocity refers to the speed of data processing. For time-sensitive processes such as catching fraud, big data must be used as it streams into your enterprise in order to maximize its value.

C. Problem with Big Data Processing

i. Heterogeneity and Incompleteness

When humans consume information, a great deal of heterogeneity is comfortably tolerated. In fact, the nuance and richness of natural language can provide valuable depth. However, machine analysis algorithms expect homogeneous data, and cannot understand nuance. In consequence, data must be carefully structured as a first step in (or prior to) data analysis. Computer systems work most efficiently if they can store multiple items that are all identical in size and structure. Efficient representation, access, and analysis of semi-structured data require further work.

ii. Scale

Of course, the first thing anyone thinks of with Big Data is its size. After all, the word "big" is there in the very name. Managing large and rapidly increasing volumes of data has been a challenging issue for many decades. In the past, this challenge was mitigated by processors getting faster, following Moore's law, to provide us with the resources needed to cope with increasing volumes of data. But, there is a fundamental shift underway now: data volume is scaling faster than compute resources, and CPU speeds are static.

iii. Timeliness

The flip side of size is speed. The larger the data set to be processed, the longer it will take to analyze. The design of a system that effectively deals with size is likely also to result in a system that can process a given size of data set faster. However, it is not just this speed that is usually meant when one speaks of Velocity in the context of Big Data. Rather, there is an acquisition rate challenge

iv. Privacy

The privacy of data is another huge concern, and one that increases in the context of Big Data. For electronic health records, there are strict laws governing what can and cannot

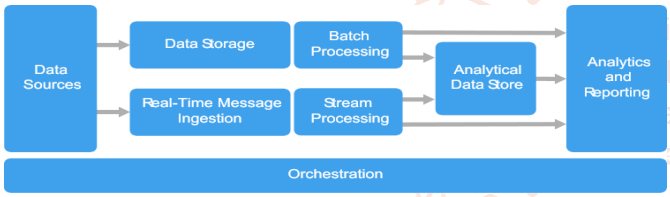
be done. For other data, regulations, particularly in the US, are less forceful. However, there is great public fear regarding the inappropriate use of personal data, particularly through linking of data from multiple sources. Managing privacy is effectively both a technical and a sociological problem, which must be addressed jointly from both perspectives to realize the promise of big data.

v. Human Collaboration

In spite of the tremendous advances made in computational analysis, there remain many patterns that humans can easily detect but computer algorithms have a hard time finding. Ideally, analytics for Big Data will not be all computational rather it will be designed explicitly to have a human in the loop. The new sub-field of visual analytics is attempting to do this, at least with respect to the modeling and analysis phase in the pipeline. In today's complex world, it often takes multiple experts from different domains to really understand what is going on. A Big Data analysis system must support input from multiple human experts, and shared exploration of results. These multiple experts may be separated in space and time when it is too expensive to assemble an entire team together in one room. The data system has to accept this distributed expert input, and support their collaboration.

D. Big data architecture style

A big data architecture is designed to handle the ingestion, processing, and analysis of data that is too large or complex for traditional database systems.



Architecture of Big Data

Big data solutions typically involve one or more of the following types of workload:

- Batch processing of big data sources at rest.
- Real-time processing of big data in motion.
- Interactive exploration of big data.
- Predictive analytics and machine learning.
- Most big data architectures include some or all of the following components:

Data sources: All big data solutions start with one or more data sources. Examples include:

- Application data stores, such as relational databases.
- Static files produced by applications, such as web server log files.
- Real-time data sources, such as IoT devices.

Data storage: Data for batch processing operations is typically stored in a distributed file store that can hold high volumes of large files in various formats. This kind of store is often called a data lake.

Batch processing: Because the data sets are so large, often a big data solution must process data files using long-running batch jobs to filter, aggregate, and otherwise prepare the data for analysis. Usually these jobs involve reading source files, processing them, and writing the output to new files

Real-time message ingestion: If the solution includes real-time sources, the architecture must include a way to capture and store real-time messages for stream processing. This might be a simple data store, where incoming messages are dropped into a folder for processing. However, many solutions need a message ingestion store to act as a buffer for messages, and to support scale-out processing, reliable delivery, and other message queuing semantics.

Stream processing: After capturing real-time messages, the solution must process them by filtering, aggregating, and otherwise preparing the data for analysis. The processed stream data is then written to an output sink.

Analytical data store: Many big data solutions prepare data for analysis and then serve the processed data in a structured format that can be queried using analytical tools.

Analysis and reporting: The goal of most big data solutions is to provide insights into the data through analysis and reporting. To empower users to analyze the data, the architecture may include a data modeling layer, such as a multidimensional

Orchestration: Most big data solutions consist of repeated data processing operations, encapsulated in workflows, that transform source data, move data between multiple sources and sinks, load the processed data into an analytical data store, or push the results straight to a report or dashboard

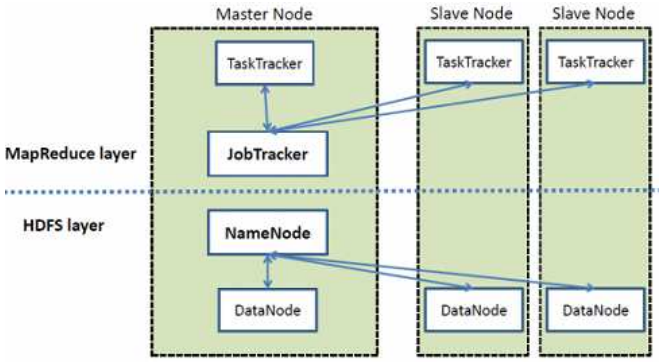
2. Hadoop: Solution for Big Data Processing

Hadoop is a Programming framework used to support the processing of large data sets in a distributed computing environment. Hadoop was developed by Google's MapReduce that is a software framework where an application break down into various parts. The Current Apache Hadoop ecosystem consists of the Hadoop Kernel, MapReduce, HDFS and numbers of various components like Apache Hive, Base and Zookeeper. HDFS and MapReduce are explained in following points.

Hadoop Architecture Overview

Apache Hadoop offers a scalable, flexible and reliable distributed computing big data framework for a cluster of systems with storage capacity and local computing power by leveraging commodity hardware. Hadoop follows a Master Slave architecture for the transformation and analysis of large datasets using Hadoop MapReduce paradigm. The 3 important hadoop components that play a vital role in the Hadoop architecture are -

a. Hadoop Distributed File System (HDFS) – Patterned after the UNIX file system
b. Hadoop MapReduce
c. Yet Another Resource Negotiator (YARN)



Hadoop Architecture

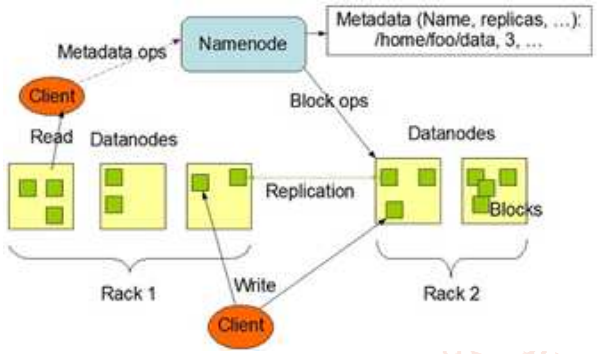
Hadoop Architecture Explained

Hadoop skillset requires thoughtful knowledge of every layer in the hadoop stack right from understanding about the various components in the hadoop architecture, designing a hadoop cluster, performance tuning it and setting up the top chain responsible for data processing.

Hadoop follows a master slave architecture design for data storage and distributed data processing using HDFS and MapReduce respectively. The master node for data storage is hadoop HDFS is the NameNode and the master node for parallel processing of data using Hadoop MapReduce is the Job Tracker. The slave nodes in the hadoop architecture are the other machines in the Hadoop cluster which store data and perform complex computations. Every slave node has a Task Tracker daemon and a DataNode that synchronizes the processes with the Job Tracker and NameNode respectively. In Hadoop architectural implementation the master or slave systems can be setup in the cloud or on-premise.

Role of Distributed Storage - HDFS in Hadoop Application Architecture Implementation

A file on HDFS is split into multiple bocks and each is replicated within the Hadoop cluster. A block on HDFS is a blob of data within the underlying file system with a default size of 64MB.The size of a block can be extended up to 256 MB based on the requirements.



HDFS Architecture

Hadoop Distributed File System (HDFS) stores the application data and file system metadata separately on dedicated servers. NameNode and DataNode are the two critical components of the Hadoop HDFS architecture. Application data is stored on servers referred to as DataNodes and file system metadata is stored on servers referred to as NameNode. HDFS replicates the file content on multiple DataNodes based on the replication factor to ensure reliability of data. The NameNode and DataNode communicate with each other using TCP based protocols. For the Hadoop architecture to be performance efficient, HDFS must satisfy certain pre-requisites –

- All the hard drives should have a high throughput.
- Good network speed to manage intermediate data transfer and block replications.

NameNode

All the files and directories in the HDFS namespace are represented on the NameNode by Inodes that contain various attributes like permissions, modification timestamp, disk space quota, namespace quota and access times. NameNode maps the entire file system structure into memory. Two files fsimage and edits are used for persistence during restarts.

- Fsimage file contains the Inodes and the list of blocks which define the metadata.It has a complete snapshot of the file systems metadata at any given point of time.
- The edits file contains any modifications that have been performed on the content of the fsimage file.Incremental changes like renaming or appending data to the file are stored in the edit log to ensure durability instead of creating a new fsimage snapshot everytime the namespace is being altered.

When the NameNode starts, fsimage file is loaded and then the contents of the edits file are applied to recover the latest state of the file system. The only problem with this is that over the time the edits file grows and consumes all the disk space resulting in slowing down the restart process. If the hadoop cluster has not been restarted for months together then there will be a huge downtime as the size of the edits file will be increase. This is when Secondary NameNode comes to the rescue. Secondary NameNode gets the fsimage and edits log from the primary NameNode at regular intervals and loads both the fsimage and edit logs file to the main memory by applying each operation from edits log file to fsimage. Secondary NameNode copies the new fsimage file to the primary NameNode and also will update the modified time of the fsimage file to fstime file to track when then fsimage file has been updated.

DataNode

DataNode manages the state of an HDFS node and interacts with the blocks .A DataNode can perform CPU intensive jobs like semantic and language analysis, statistics and machine learning tasks, and I/O intensive jobs like clustering, data import, data export, search, decompression, and indexing. A DataNode needs lot of I/O for data processing and transfer.

On startup every DataNode connects to the NameNode and performs a handshake to verify the namespace ID and the software version of the DataNode. If either of them does not match then the DataNode shuts down automatically. A DataNode verifies the block replicas in its ownership by sending a block report to the NameNode. As soon as the DataNode registers, the first block report is sent. DataNode sends heartbeat to the NameNode every 3 seconds to confirm that the DataNode is operating and the block replicas it hosts are available.

Role of Distributed Computation - MapReduce in Hadoop Application Architecture Implementation

The heart of the distributed computation platform Hadoop is its java-based programming paradigm Hadoop MapReduce. Map or Reduce is a special type of directed acyclic graph that can be applied to a wide range of business use cases. Map function transforms the piece of data into key-value pairs and then the keys are sorted where a reduce function is applied to merge the values based on the key into a single output.

How does the Hadoop MapReduce architecture work?

The execution of a MapReduce job begins when the client submits the job configuration to the Job Tracker that specifies the map, combine and reduce functions along with the location for input and output data. On receiving the job configuration, the job tracker identifies the number of splits based on the input path and select Task Trackers based on their network vicinity to the data sources. Job Tracker sends a request to the selected Task Trackers.

The processing of the Map phase begins where the Task Tracker extracts the input data from the splits. Map function is invoked for each record parsed by the "InputFormat" which produces key-value pairs in the memory buffer. The memory buffer is then sorted to different reducer nodes by invoking the combine function. On completion of the map task, Task Tracker notifies the Job Tracker. When all Task Trackers are done, the Job Tracker notifies the selected Task Trackers to begin the reduce phase. Task Tracker reads the region files and sorts the key-value pairs for each key. The reduce function is then invoked which collects the aggregated values into the output file.

Hadoop Architecture Design – Best Practices to Follow

Use good-quality commodity servers to make it cost efficient and flexible to scale out for complex business use cases. One of the best configurations for Hadoop architecture is to begin with 6 core processors, 96 GB of memory and 1 0 4 TB of local hard drives. This is just a good configuration but not an absolute one.

- For faster and efficient processing of data, move the processing in close proximity to data instead of separating the two.
- Hadoop scales and performs better with local drives so use Just a Bunch of Disks (JBOD) with replication instead of redundant array of independent disks (RAID).
- Design the Hadoop architecture for multi-tenancy by sharing the compute capacity with capacity scheduler and share HDFS storage.
- Do not edit the metadata files as it can corrupt the state of the Hadoop cluster.

3. Literature Review

S. Vikram Phaneendra & E. Madhusudhan Reddy et.al. Illustrated that in olden days the data was less and easily handled by RDBMS but recently it is difficult to handle huge data through RDBMS tools, which is preferred as "big data". In this they told that big data differs from other data in 5 dimensions such as volume, velocity, variety, value and complexity. They illustrated the hadoop architecture consisting of name node, data node, edge node, HDFS to handle big data systems. Hadoop architecture handle large data sets, scalable algorithm does log management application of big data can be found out in financial, retail industry, health-care, mobility, insurance. The authors also focused on the challenges that need to be faced by enterprises when handling big data: - data privacy, search analysis, etc [1].

Kiran kumara Reddi & Dnvsl Indira et.al. Enhanced us with the knowledge that Big Data is combination of structured , semi-structured ,unstructured homogenous and heterogeneous data .The author suggested to use nice model to handle transfer of huge amount of data over the network .Under this model, these transfers are relegated to low demand periods where there is ample ,idle bandwidth available . This bandwidth can then be repurposed for big data transmission without impacting other users in system. The Nice model uses a store –and-forward approach by utilizing staging servers. The model is able to accommodate differences in time zones and variations in bandwidth. They suggested that new algorithms are required to transfer big data and to solve issues like security, compression, routing algorithms [2].

Jimmy Lin et.al. used Hadoop which is currently the large – scale data analysis " hammer" of choice, but there exists classes of algorithms that aren't " nails" in the sense that they are not particularly amenable to the MapReduce programming model . He focuses on the simple solution to find alternative non-iterative algorithms that solves the same problem. The standard MapReduce is well known and described in many places .Each iteration of the pagerank corresponds to the MapReduce job. The author suggested iterative graph, gradient descent & EM iteration which is typically implemented as Hadoop job with driven set up iteration &Check for convergences. The author suggests that if all you have is a hammer, throw away everything that's not a nail [3].

Wei Fan & Albert Bifet et.al. Introduced Big Data Mining as the capability of extracting Useful information from these large datasets or streams of data that due to its Volume, variability and velocity it was not possible before to do it. The author also started that there are certain controversy about Big Data. There certain tools for processes. Big Data as such hadoop, strom, apache S4. Specific tools for big graph mining were PEGASUS & Graph. There are certain Challenges that need to death with as such compression, visualization etc.[4].

Albert Bifet et.al. Stated that streaming data analysis in real time is becoming the fastest and most efficient way to obtain useful knowledge, allowing organizations to react quickly when problem appear or detect to improve performance. Huge amount of data is created everyday termed as " big data". The tools used for mining big data are apache hadoop, apache big, cascading, scribe, storm, apache hbase, apache mahout, MOA, R, etc. Thus, he instructed that our ability to handle many exabytes of data mainly dependent on existence of rich variety dataset, technique, software framework [5].

Bernice Purcell et.al. Started that Big Data is comprised of large data sets that can't be handle by traditional systems. Big data includes structured data, semi-structured and unstructured data. The data storage technique used for big data includes multiple clustered network attached storage (NAS) and object based storage. The Hadoop architecture is used to process unstructured and semi-structured using map reduce to locate all relevant data then select only the data directly answering the query. The advent of Big Data has posed opportunities as well challenges to business [6].

Sameer Agarwal et.al. Presents a BlinkDB, a approximate query engine for running interactive SQL queries on large volume of data which is massively parallel. BlinkDB uses two key ideas: (1) an adaptive optimization framework that builds and maintains a set of multi-dimensional stratified samples from original data over time, and (2) A dynamic sample selection strategy that selects an appropriately sized sample based on a query's accuracy or response time requirements [7].

Yingyi Bu et.al. Used a new technique called as HaLoop which is modified version of Hadoop MapReduce Framework, as Map Reduce lacks built-in-support for iterative programs HaLoop allows iterative applications to be assembled from existing Hadoop programs without modification, and significantly improves their efficiency by providing inter- iteration caching mechanisms and a loop-

aware scheduler to exploit these caches. He presents the design, implementation, and evaluation of HaLoop, a novel parallel and distributed system that supports large-scale iterative data analysis applications. HaLoop is built on top of Hadoop and extends it with a new programming model and several important optimizations that include (1) a loop-aware task scheduler, (2) loop-invariant data caching, and (3) caching for efficient fix point verification [8].

Shadi Ibrahim et.al. Project says presence of partitioning skew1 causes a huge amount of data transfer during the shuffle phase and leads to significant unfairness on the reduce input among different data nodes In this paper, author develop a novel algorithm named LEEN for locality aware and fairness-aware key partitioning in MapReduce. LEEN embraces an asynchronous map and reduce scheme. Author has integrated LEEN into Hadoop. His experiments demonstrate that LEEN can efficiently achieve higher locality and reduce the amount of shuffled data. More importantly, LEEN guarantees fair distribution of the reduce inputs. As a result, LEEN achieves a performance improvement of up to 45% on different workloads. To tackle all this he presents a present a technique for Handling Partitioning Skew in MapReduce using LEEN [9].

Kenn Slagter et.al. Proposes an improved partitioning algorithm that improves load balancing and memory consumption. This is done via an improved sampling algorithm and partitioner. To evaluate the proposed algorithm, its performance was compared against a state of the art partitioning mechanism employed by Tera Sort as the performance of MapReduce strongly depends on how evenly it distributes this workload. This can be a challenge, especially in the advent of data skew. In MapReduce, workload distribution depends on the algorithm that partitions the data. One way to avoid problems inherent from data skew is to use data sampling. How evenly the partitioner distributes the data depends on how large and representative the sample is and on how well the samples are analyzed by the partitioning mechanism. He uses an improved partitioning mechanism for optimizing massive data analysis using MapReduce for evenly distribution of workload [10].

Ahmed Eldawy et.al. presents the first full-fledged MapReduce framework with native support for spatial data that is spatial data Spatial Hadoop pushes its spatial constructs in all layers of Hadoop, namely, language, storage, MapReduce and operations layers. In the language layer, a simple high level language is provided to simplify spatial data analysis for non-technical users. In the storage layer, a two-layered spatial index structure is provided where the global index partitions data across nodes while the local index organizes data in each node. This structure is used to build a grid index, an R-tree or an R+-tree. Spatial-Hadoop is a comprehensive extension to Hadoop that pushes spatial data inside the core functionality of Hadoop. Spatial Hadoop runs existing Hadoop programs as is, yet, it achieves order(s) of magnitude better performance than Hadoop when dealing with spatial data. SpatialHadoop employs a simple spatial high level language, a two-level spatial index structure, basic spatial components built inside the MapReduce layer, and three basic spatial operations: range queries, k-NN queries, and spatial join. Author presents an efficient MapReduce framework for Spatial Data [11].

Jeffrey Dean et.al. Implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers and the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine Communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. Author proposes Simplified Data Processing on Large Clusters [12].

Chris Jermaine et. al. Proposes a Online Aggregation for Large-Scale Computing. Given the potential for OLA to be newly relevant, and given the current interest on very large-scale, data-oriented computing, in this paper we consider the problem of providing OLA in a shared-nothing environment. While we concentrate on implementing OLA on top of a MapReduce engine, many of author's most basic project contributions are not specific to MapReduce, and should apply broadly. Consider how online aggregation can be built into a MapReduce system for large-scale data processing. Given the MapReduce paradigm's close relationship with cloud computing (in that one might expect a large fraction of MapReduce jobs to be run in the cloud), online aggregation is a very attractive technology. Since large-scale cloud computations are typically pay-as-you-go, a user can monitor the accuracy obtained in an online fashion, and then save money by killing the computation early once sufficient accuracy has been obtained [13].

Tyson Condie et.al. propose a modified MapReduce architecture in which intermediate data is pipelined between operators, while preserving the programming interfaces and fault tolerance models of other MapReduce frameworks. To validate this design, author developed the Hadoop Online Prototype (HOP), a pipelining version of Hadoop. Pipelining provides several important advantages to a MapReduce framework, but also raises new design challenges. To simplify fault tolerance, the output of each MapReduce task and job is materialized to disk before it is consumed. In this demonstration, we describe a modified MapReduce architecture that allows data to be pipelined between operators. This extends the MapReduce programming model beyond batch processing, and can reduce completion times and improve system utilization for batch jobs as well. We demonstrate a modified version of the Hadoop MapReduce framework that supports online aggregation, which allows users to see "early returns" from a job as it is being computed. Our Hadoop Online Prototype (HOP) also supports continuous queries, which enable MapReduce programs to be written for applications such as event monitoring and stream processing [14].

Jonathan Paul Olmsted et.al. Derive the necessary results to apply variation Bayesian inference to the ideal point model. This deterministic, approximate solution is shown to produce comparable results to those from standard estimation strategies. However, unlike these other

estimation approaches, solving for the (approximate) posterior distribution is rapid and easily scales to 'big data'. Inferences from the variation Bayesian approach to ideal point estimation are shown to be equivalent to standard approaches on modestly-sized roll call matrices from recent sessions of the US Congress. Then, the ability of variation inference to scale to big data is demonstrated and contrasted with the performance of standard approaches.[15]

Jonathan Stuart Ward et.al. did a survey of Big data definition, Anecdotally big data is predominantly associated with two ideas: data storage and data analysis. Despite the sudden Interest in big data, these concepts are far from new and have long lineages. This, therefore, raises the question as to how big data is notably different from conventional data processing techniques. For rudimentary insight as to the answer to this question one need look no further than the term big data. \Big" implies significance, complexity and challenge. Unfortunately the term\big" also invites quantification and therein lies the difficulty in furnishing a definition. The lack of a consistent definition introduces ambiguity and hampers discourse relating to big data. This short paper attempts to collate the various definitions which have gained some degree of traction and to furnish a clear and concise definition of an otherwise ambiguous term [16].

Albert Bifet et.al. Discuss the current and future trends of mining evolving data streams, and the challenges that the field will have to overcome during the next years. Data stream real time analytics are needed to manage the data currently generated, at an ever increasing rate, from such applications as: sensor networks, measurements in network monitoring and traffic management, log records or click-streams in web exploring, manufacturing processes, call detail records, email, blogging, twitter posts and others. In fact, all data generated can be considered as streaming data or as a snapshot of streaming data, since it is obtained from an interval of time. Streaming data analysis in real time is becoming the fastest and most efficient way to obtain useful knowledge from what is happening now, allowing organizations to react quickly when problems appear or to detect new trends helping to improve their performance. Evolving data streams are contributing to the growth of data created over the last few years. We are creating the same quantity of data every two days, as we created from the dawn of time up until 2003. Evolving data streams methods are becoming a low-cost, green methodology for real time online prediction and analysis [17].

4. Conclusion

We have entered an era of Big Data. The paper describes the concept of Big Data along with 3 Vs, Volume, Velocity and variety of Big Data. The paper also focuses on Big Data processing problems. These technical challenges must be addressed for efficient and fast processing of Big Data. The challenges include not just the obvious issues of scale, but also heterogeneity, lack of structure, error-handling, privacy, timeliness, provenance, and visualization, at all stages of the analysis pipeline from data acquisition to result interpretation. These technical challenges are common across a large variety of application domains, and therefore not cost-effective to address in the context of one domain alone. The paper describes Hadoop which is an open source software used for processing of Big Data.

REFERENCES

[1] S. Vikram Phaneendra & E. Madhusudhan Reddy "Big Data- solutions for RDBMS problems- A survey" In 12th IEEE/IFIP Network Operations & Management Symposium (NOMS 2010) (Osaka, Japan, Apr 19{23 2013).

[2] Kiran kumara Reddi & Dnvsl Indira "Different Technique to Transfer Big Data : survey" IEEE Transactions on 52(8) (Aug.2013) 2348 { 2355}

[3] Jimmy Lin "MapReduce Is Good Enough?" The control project. IEEE Computer 32 (2013).

[4] Umasri.M.L, Shyamalagowri.D ,Suresh Kumar.S "Mining Big Data:- Current status and forecast to the future" Volume 4, Issue 1, January 2014 ISSN: 2277 128X

[5] Albert Bifet "Mining Big Data In Real Time" Informatica 37 (2013) 15–20 DEC 2012

[6] Bernice Purcell "The emergence of "big data" technology and analytics" Journal of Technology Research 2013.

[7] Sameer Agarwal†, Barzan MozafariX, Aurojit Panda†, Henry Milner†, Samuel MaddenX, Ion Stoica "BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data" Copyright © 2013ì ACM 978-1-4503-1994 2/13/04

[8] Yingyi Bu _ Bill Howe _ Magdalena Balazinska _ Michael D. Ernst "The HaLoop Approach to Large-Scale Iterative Data Analysis" VLDB 2010 paper "HaLoop: Efficient Iterative Data Processing on Large Clusters.

[9] Shadi Ibrahim⋆ _ Hai Jin _ Lu Lu "Handling Partitioning Skew in MapReduce using LEEN" ACM 51 (2008) 107–113

[10] Kenn Slagter · Ching-Hsien Hsu "An improved partitioning mechanism for optimizing massive data analysis using MapReduce" Published online: 11 April 2013©Springer Science+Business Media New York 2013.

[11] Ahmed Eldawy, Mohamed F. Mokbel "A Demonstration of SpatialHadoop:An Efficient MapReduce Framework for Spatial Data" Proceedings of the VLDB Endowment, Vol. 6, No. 12 Copyright 2013 VLDB Endowment 21508097/13/10.

[12] Jeffrey Dean and Sanjay Ghemawat "MapReduce: Simplified Data Processing on Large Clusters" OSDI 2010

[13] Niketan Pansare1, Vinayak Borkar2, Chris Jermaine1, Tyson Condie "Online Aggregation for Large MapReduce Jobs" August 29September 3, 2011, Seattle, WA Copyright 2011 VLDB Endowment, ACM

[14] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein "Online Aggregation and Continuous Query support in MapReduce" SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA. Copyright 2010 ACM 978-1-4503-0032-2/10/06.

[15] Jonathan Paul Olmsted "Scaling at Scale: Ideal Point Estimation with 'Big-Data" Princeton Institute for Computational Science and Engineering 2014.

[16] Jonathan Stuart Ward and Adam Barker "Undefined By Data: A Survey of Big Data Definitions" Stamford, CT: Gartner, 2012.

[17] Balaji Palanisamy, Member, IEEE, Aameek Singh, Member, IEEE Ling Liu, Senior Member, IEEE" Cost-effective Resource Provisioning for MapReduce in a Cloud"gartner report 2010, 25