ISSN:0975-9646

Survey Paper On Big Data

Ms. Vibhavari Chavan, Prof. Rajesh. N. Phursule

Department of Computer Engineering
JSPM's Imperial College of Engineering and Research, Pune

Abstract- **Big data is the term for any collection of data sets so large and complex that it becomes difficult to process using traditional data processing applications. The challenges include analysis, capture, curation, search, sharing, storage, transfer, visualization, and privacy violations. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, prevent diseases, combat crime and so on." Big data is difficult to work with using most relational database management systems and desktop statistics and visualization packages, requiring instead "massively parallel software running on tens, hundreds, or even thousands of servers". Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time. Big data "size" is a constantly moving target, as of its ranging from a few dozen terabytes to many petabytes of data. Big data is a set of techniques and technologies that require new forms of integration to uncover large hidden values from large datasets that are diverse, complex, and of a massive scale. Big data environment is used to acquire, organize and analyze the various types of data.There is an observation about Map Reduce framework that framework generates large amount of intermediate data. Therefore, as well as the tasks finishes there is need of throwing that abundant data, because MapReduce is unable to utilize them.**

Index Terms- **Big data, Hadoop, HDFS, MapReduce, Pig, Hive, Hbase,**

I.INTRODUCTION

1. WHAT IS BIG DATA?

We create 2.5 quintillion bytes of data — so much that 90% of the data in the world today has been created in the last two years alone. This much amount of data comes from everywhere: sensors used to gather climate information, posts to social media sites, digital pictures and videos, purchase transaction records, and cell phone GPS signals to name a few. This huge amount of the data is known as "**Big data"**[14]. Big data is a buzzword, or catch-phrase, utilizes to describe a massive volume of both structured and unstructured data that is so huge that it's complicated to process using traditional database and software techniques. In most enterprise scenarios the data is too large or it moves too fast or it exceeds current processing capacity. Big data has the potential to help organizations to improve operations and make faster, more intelligent decisions[15]. **Big Data**, now a days this term becomes common in IT industries. As there is a huge amount of data lies in the industry but there is nothing before big data comes into picture [3]. Big data is actually an evolving term that describes any voluminous amount of structured, semi-structured and unstructured data that has the potential to be mined for information. Although big data doesn't refer to any specific quantity, so this term is often used when speaking about petabytes and exabytes of data[16]. **Big data** is an all-encompassing term for large collection of the data sets so this huge and complex that it becomes difficult to operate them using traditional data processing applications. When dealing with larger datasets, organizations face difficulties in being able to create, manipulate, and manage big data. Big data is particularly a problem in business analytics because standard tools and procedures are not designed to search and analyze massive datasets.

An example of big data might be petabytes (1,024 terabytes) or exabytes (1,024 petabytes) of data consisting of billions to trillions of records of millions of people—all from different sources (e.g. Web, sales, customer contact center, social media, mobile data and so on). The data is typically loosely structured data that is often incomplete and inaccessible[15].

The challenges include analysis, capture, curation, search, sharing, storage, transfer, visualization, and privacy violations. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, prevent diseases, combat crime and so on"[10].Scientists regularly encounter limitations due to large data sets in many areas, including meteorology, genomics, connectomics, complex physics simulations, and biological and environmental research. The limitations also affect Internet search, finance and business informatics. Data sets grow in size in part because they are increasingly being gathered by ubiquitous information-sensing mobile devices, aerial sensory technologies (remote sensing), software logs, cameras, microphones, radio-frequency identification (RFID) readers, and wireless sensor networks. The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s;as of 2012, every day 2.5 exabytes (2.5×10^{18}) of data were created. The challenge for large enterprises is determining who should own big data initiatives that straddle the entire organization.

Big data defined as far back as 2001, industry analyst Doug Laney (currently with Gartner) articulated the now mainstream definition of big data as the three Vs of big data: volume, velocity and variety [18]. Big data can be characterized by welknown 3Vs: the extreme volume of data, the wide variety of types of data and the velocity at which the data must be must processed. Although big data doesn't refer to any specific quantity, the term is often used when speaking about petabytes and exabytes of data, much of which cannot be integrated easily. [16]

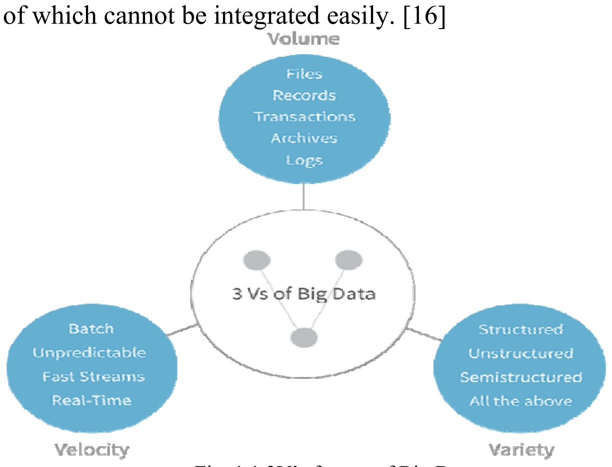

Fig. 1.1 3V's factors of Big Data

- **Volume.** Many factors contribute to the increase in data volume. Transaction-based data stored through the years. Unstructured data streaming in from social media. Increasing amounts of sensor and machine-to-machine data being collected. In the past, excessive data volume was a storage issue. But with decreasing storage costs, other issues emerge, including how to determine relevance within large data volumes and how to use analytics to create value from relevant data.fig.1.1
- **Velocity.** Data is streaming in at unprecedented speed and must be dealt with in a timely manner. RFID tags, sensors and smart metering are driving the need to deal with torrents of data in near-real time. Reacting quickly enough to deal with data velocity is a challenge for most organizations.fig.1.1
- **Variety.** Data today comes in all types of formats. Structured, numeric data in traditional databases. Information created from line-of-business applications. Unstructured text documents, email, video, audio, stock ticker data and financial transactions. Managing, merging and governing different varieties of data is something many organizations still grapple with.fig.1.1

We consider two additional dimensions when thinking about big data:

- **Variability.** In addition to the increasing velocities and varieties of data, data flows can be highly inconsistent with periodic peaks. Is something trending in social media? Daily, seasonal and event-triggered peak data loads can be challenging to manage. Even more so with unstructured data involved.
- **Complexity.** Today's data comes from multiple sources. And it is still an undertaking to link, match, cleanse and transform data across systems. However, it is necessary to connect and correlate relationships, hierarchies and multiple data linkages or your data can quickly spiral out of control [17].

Data storage has grown significantly, shifting makedly from analog to digital after 2000 fig.1.2

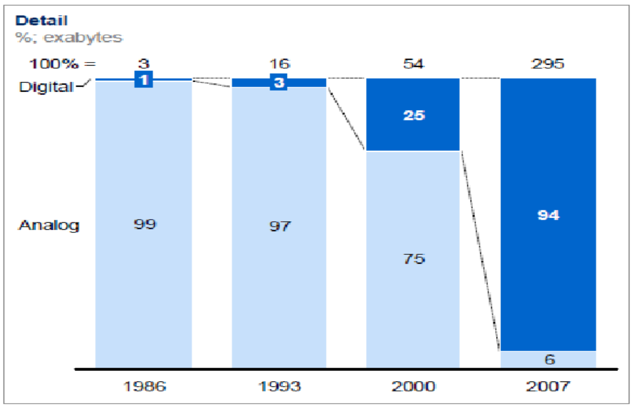

Fig. 1.2 Data storage growth graph

Big Data, the analysis of huge quantities of data to gain new insight has become a ubiquitous phrase in recent years. As we know that day by day the data is growing at a staggering rate. One of the efficient well-known technologies that deal with the Big Data is Hadoop [6].

2. Hadoop:

Hadoop was created by Doug Cutting and Mike Cafarella in 2005. Doug Cutting, who was working at Yahoo! at the time, named it after his son's toy elephant. It was originally developed to support distribution for the Nutch search engine project. Hadoop is open-source software that enables reliable, scalable, distributed computing on clusters of inexpensive servers[1].

Hadoop is:

- *Reliable*: The software is fault tolerant, it expects and handles hardware and software failures
- *Scalable*: Designed for massive scale of processors, memory, and local attached storage
- *Distributed*: Handles replication. Offers massively parallel programming model, Map Reduce

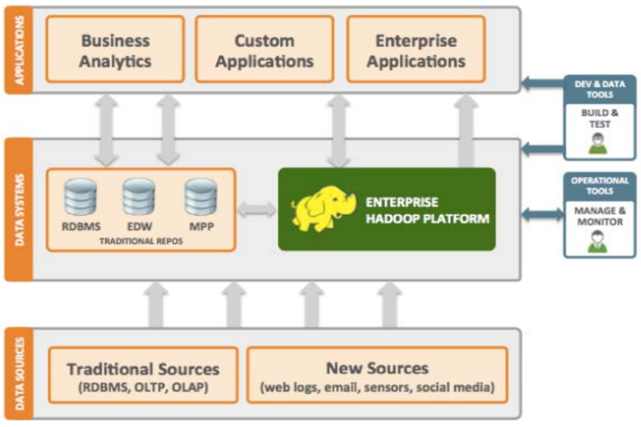

Fig 2.1 Hadoop system

Hadoop is an Open Source implementation of a large-scale batch processing system. That use the Map-Reduce framework introduced by Google by leveraging the concept of map and reduce functions that well known used in Functional Programming. Although the Hadoop framework is written in Java, it allows developers to deploy custom-written programs coded in Java or any other language to process data in a parallel fashion across hundreds or thousands of commodity servers. It is optimized for contiguous read requests(streaming reads), where processing includes of scanning all the data. Depending on the complexity of the process and the volume of data, response time can vary from minutes to hours. While Hadoop can processes data fast, so its key advantage is its massive scalability [20].

Hadoop is currently being used for index web searches, email spam detection, recommendation engines, prediction in financial services, genome manipulation in life sciences, and for analysis of unstructured data such as log, text, and clickstream. While many of these applications could in fact be implemented in a relational database(RDBMS)fig 2.1, the main core of the Hadoop framework is functionally different from an RDBMS. The following discusses some of these differences Hadoop is particularly useful when:

- Complex information processing is needed
- Unstructured data needs to be turned into structured data
- Queries can’t be reasonably expressed using SQL
- Heavily recursive algorithms
- Complex but parallelizable algorithms needed, such as geo-spatial analysis or genome sequencing
- Machine learning
- Data sets are too large to fit into database RAM, discs, or require too many cores (10’s of TB up to PB)
- Data value does not justify expense of constant real-time availability, such as archives or special interest info, which can be moved to Hadoop and remain available at lower cost
- Results are not needed in real time
- Fault tolerance is critical
- Significant custom coding would be required to handle job scheduling

Hadoop was inspired by Google's MapReduce, a software framework in which an application is broken down into numerous small parts. Any of these parts (also called fragments or blocks) can be run on any node in the cluster. Doug Cutting, Hadoop's creator, named the framework after his child's stuffed toy elephant. The current Apache Hadoop ecosystem consists of the Hadoop kernel, MapReduce, the Hadoop distributed file system (HDFS) and a number of related projects such as Apache Hive, HBase and Zookeeper. The Hadoop framework is used by major players including Google, Yahoo and IBM, largely for applications involving search engines and advertising. The preferred operating systems are Windows and Linux but Hadoop can also work with BSD and OS X [21].

A distributed file system is a client/server-based application that allows clients to access and process data stored on the server as if it were on their own computer. A distributed file system is a client/server-based application that allows clients to access and process data stored on the server as if it were on their own computer. When a user accesses a file on the server, the server sends the user a copy of the file, which is cached on the user's computer while the data is being processed and is then returned to the server. Ideally, a distributed file system organizes file and directory services of individual servers into a global directory in such a way that remote data access is not location-specific but is identical from any client. All files are accessible to all users of the global file system and organization is hierarchical and directory-based[2].

Since more than one client may access the same data simultaneously, the server must have a mechanism in place (such as maintaining information about the times of access) to organize updates so that the client always receives the most current version of data and that data conflicts do not arise. Distributed file systems typically use file or database replication (distributing copies of data on multiple servers) to protect against data access failures[4].Sun Microsystems' Network File System (NFS)[21],Novell NetWare, Microsoft's Distributed File System, and IBM/Transarc's DFS are some examples of distributed file systems.

II. HDFS

The Hadoop Distributed File System (HDFS) is the file system component of the Hadoop framework[13]. HDFS is designed and optimized to store data over a large amount of low-cost hardware in a distributed fashion.

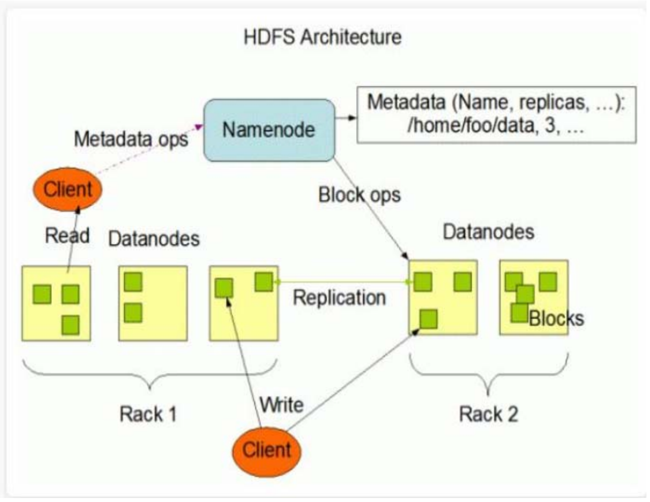

Fig.3.1 HDFS architecture

Name Node :

Name node is a type of the master node, which is having the information that means meta data about the all data node there is address(use to talk), free space, data they store, active data node , passive data node, task tracker, job tracker and many other configuration such as replication of data [3].

The NameNode records all of the metadata, attributes, and locations of files and data blocks in to the DataNodes. The attributes it records are the things like file permissions, file modification and access times, and namespace, which is a hierarchy of files and directories. The NameNode maps the namespace tree to file blocks in DataNodes. When a client node wants to read a file in the HDFS it first contacts the Namenode to receive the location of the data blocks

associated with that file [27].
A NameNode stores information about the overall system because it is the master of the HDFS with the DataNodes being the slaves. It stores the image and journal logs of the system. The image of the system is a list of blocks and data for each file stored in the HDFS. The journal is just a modification log of the image. The NameNode must always store the most up to date image and journal. Basically, the NameNode always knows where the data blocks and replicates are for each file and it also knows where the free blocks are in the system so it keeps track of where future files can be written.

Data Node:
Data node is a type of slave node in the hadoop, which is used to save the data and there is task tracker in data node which is use to track on the ongoing job on the data node and the jobs which coming from name node[3].
The DataNodes store the blocks and block replicas of the file system. During startup each DataNode connects and performs a handshake with the NameNode. The DataNode checks for the accurate namespace ID, and if not found then the DataNode automatically shuts down. New DataNodes can join the cluster by simply registering with the NameNode and receiving the namespace ID [27]. Each DataNode keeps track of a block report for the blocks in its node. Each DataNode sends its block report to the NameNode every hour so that the NameNode always has an up to date view of where block replicas are located in the cluster.During the normal operation of the HDFS, each DataNode also sends a heartbeat to the NameNode every ten minutes so that the NameNode knows which DataNodes are operating correctly and are available. If after ten minutes the NameNode doesn’t receive a heartbeat from a DataNode then the NameNode assumes that the DataNode is lost and begins creating replicas of that DataNode’s lost blocks on other DataNodes. The nice thing about the HDFS architecture is that the NameNode doesn’t have to reach out to the DataNodes, it instead waits for the DataNodes to send their block reports and heartbeats to it. The NameNode can receive thousands of DataNode’s heartbeats every second and not adversely affect other NameNode operations [27].
Apache Hadoop is an open-source software framework for distributed storage and distributed processing of Big Data on clusters of commodity hardware. Its Hadoop Distributed File System (HDFS) splits files into large blocks (default 64MB or 128MB) and distributes the blocks amongst the nodes in the cluster. For processing the data, the Hadoop Map/Reduce ships code (specifically Jar files) to the nodes that have the required data, and the nodes then process the data in parallel. This approach takes advantage of data locality, in contrast to conventional HPC architecture which usually relies on a parallel file system (compute and data separated, but connected with high-speed networking).
The base Apache Hadoop framework is composed of the following modules:

- Hadoop Common – contains libraries and utilities needed by other Hadoop modules.
- Hadoop Distributed File System (HDFS) – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.
- Hadoop MapReduce – a programming model for large scale data processing.

All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are common and thus should be automatically handled in software by the framework. Apache Hadoop's MapReduce and HDFS components originally derived respectively from Google's MapReduce and Google File System (GFS) papers."Hadoop" often refers not to just the base Hadoop package but rather to the **Hadoop Ecosystem** fig.3.2, which includes all of the additional software packages that can be installed on top of or alongside Hadoop, such as Apache Hive, Apache Pig and Apache Spark.

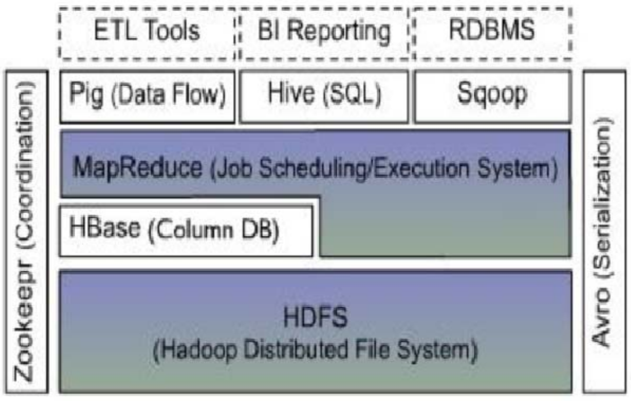

Fig. 3.2 Hadoop Ecosystem

III. Map Reduce Framework

Map Reduce is a software framework for distributed processing of large data sets on computer clusters. It is first developed by Google .Map Reduce is intended to facilitate and simplify the processing of vast amounts of data in parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner [4].
MapReduce is the key algorithm that the Hadoop MapReduce engine uses to distribute work around a cluster. Typical Hadoop cluster integrates MapReduce and HFDS layer. In MapReduce layer job tracker assigns tasks to the task tracker.Master node job tracker also assigns tasks to the slave node task tracker fig.4.1

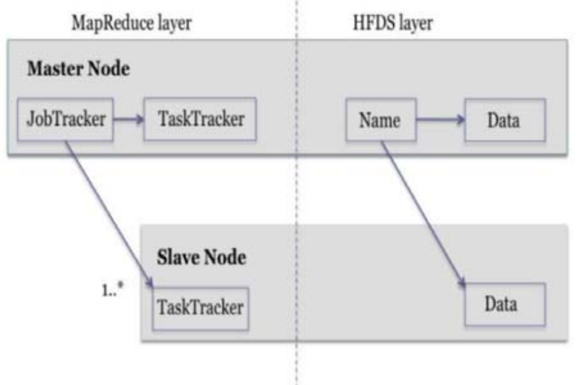

Fig. 4.1 Map reduce is based on the Maser-Slave architecture

Master node contains -
- Job tracker node (MapReduce layer)
- Task tracker node (MapReduce layer)
- Name node (HFDS layer)
- Data node (HFDS layer)

Multiple slave nodes contain -
- Task tracker node (MapReduce layer)
- Data node (HFDS layer)
- MapReduce layer has job and task tracker nodes
- HFDS layer has name and data nodes

Single JobTracker per master is responsible for scheduling the jobs' component tasks on the slaves .It monitors slave progress. It also re-executing failed tasks .As well as single TaskTracker per slave execute the tasks as directed by the master.Map reduce core functionality is based on the Map phase and reduce phase. Code usually written in Java- though it can be written in other languages with the Hadoop Streaming API.

A. Map Reduce core functionality(I):
In this functionality Map and Reduce pieces are playing vital role.
I. Map step
In this Map phase the Master node takes large problem input and slices it into smaller sub problems; distributes these to worker nodes.Worker node may do this again; leads to a multi-level tree structure .Worker processes smaller problem and hands back to master
A map transform is provided to transform an input data row of key and value to an output key/value:
- map(key1,value) -> list<key2,value2>

That is, for an input it returns a list containing zero or more (key,value) pairs:
- The output can be a different key from the input
- The output can have multiple entries with the same key

II. Reduce step:
In this Reduce phase Master node takes the answers to the sub problems and combines them in a predefined way to get the output/answer to original problem
A reduce transform is provided to take all values for a specific key, and generate a new list of the *reduced* output.
- reduce(key2, list<value2>) -> list<value3>

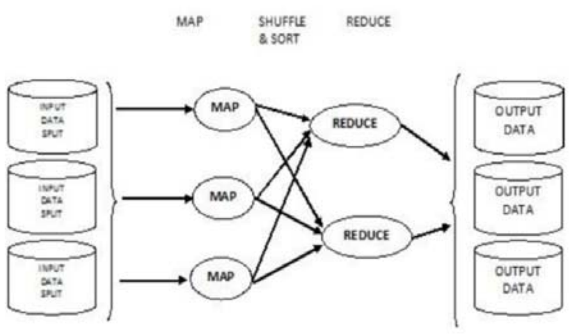

Fig. 4.2 MapReduce operation

- **"Map" step:** Each worker node applies the "map()" function to the local data, and writes the output to a temporary storage. A master node orchestrates that for redundant copies of input data, only one is processed.
- **"Shuffle" step:** Worker nodes redistribute data based on the output keys (produced by the "map()" function), such that all data belonging to one key is located on the same worker node fig.4.2.
- **"Reduce" step:** Worker nodes now process each group of output data, per key, in parallel.

B. Map Reduce core functionality(II):
Data flow beyond the two key pieces (map and reduce):

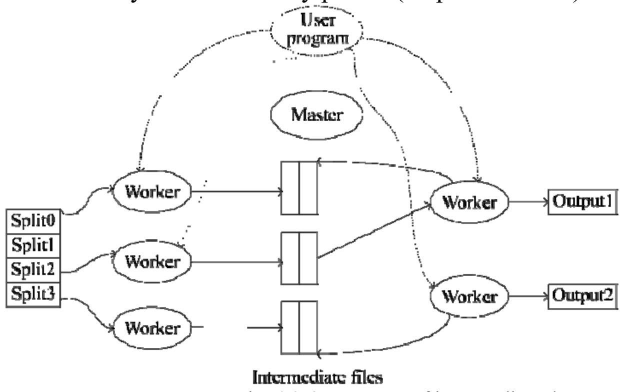

Fig. 4.3 Occurences of intermediate data

- Input reader – divides input into appropriate size splits which get assigned to a Map function
- Map function – maps file data to smaller, intermediate <key, value> pairs
- Partition function – finds the correct reducer: given the key and number of reducers, returns the desired Reduce node
- Compare function – input for Reduce is pulled from the Map intermediate output and sorted according to this compare function
- Reduce function – takes intermediate values and reduces to a smaller solution handed back to the framework
- Output writer – writes file output

C. Map Reduce core functionality(III):

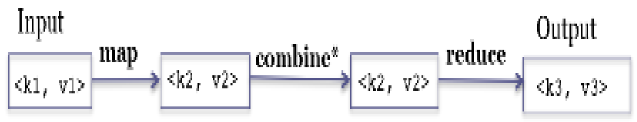

Fig 4.4 key values in MapReduce

MapReduce operates exclusively on <key, value> pairs
•Job Input: <key, value> pairs
•Job Output: <key, value> pairs

Conceivably of different types Key and value classes have to be serializable by the framework. Default serialization requires keys and values to implement Writable Key classes must facilitate sorting by the framework .

To explain in detail, we'll use a code example: WordCount
We will count occurrences of each word across different files

----

Two input files:
file1: “hello world hello moon”
file2: “goodbye world goodnight moon”

Three operations:

- ❖ Map
- ❖ Combine
- ❖ Reduce

Q. What is the output per step?

MAP

First map:	Second map:
< hello, 1 >	< goodbye, 1 >
< world, 1 >	< world, 1 >
< hello, 1 >	< goodnight, 1 >
< moon, 1 >	< moon, 1 >

COMBINE

First map:	Second map:
< moon, 1 >	< goodbye, 1 >
< world, 1 >	< world, 1 >
< hello, 2 >	< goodnight, 1 >
	< moon, 1 >

REDUCE

< goodbye, 1 >
< goodnight, 1 >
< moon, 2 >
< world, 2 >
< hello, 2 >

IV. PIG

Pig was initially developed at Yahoo! to allow people using Hadoop® to focus more on analyzing large data sets and spend less time having to write mapper and reducer programs. Like actual pigs, who eat almost anything, the Pig programming language is designed to handle any kind of data—hence the name! Stay on top of all the changes including, Hadoop-based analytics, streaming analytics, warehousing (including BigSQL), data asset discovery, integration, and governance fig.3.2.

Pig is made up of two components: the first is the language itself, which is called PigLatin[16] (people naming various Hadoop projects do tend to have a sense of humor associated with their naming conventions), and the second is a runtime environment where PigLatin programs are executed. Think of the relationship between a Java Virtual Machine (JVM) and a Java application. In this section, we’ll just refer to the whole entity as Pig. Let’s first look at the programming language itself so that you can see how it’s significantly easier than having to write mapper and reducer programs.

1. The first step in a Pig program is to LOAD the data you want to manipulate from HDFS.
2. Then you run the data through a set of transformations (which, under the covers, are translated into a set of mapper and reducer tasks).
3. Finally, you DUMP the data to the screen or you STORE the results in a file somewhere.

LOAD : As is the case with all the Hadoop features, the objects that are being worked on by Hadoop are stored in HDFS. In order for a Pig program to access this data, the program must first tell Pig what file (or files) it will use, and that’s done through the LOAD 'data_file' command (where 'data_file' specifies either an HDFS file or directory).
If a directory is specified, all the files in that directory will be loaded into the program. If the data is stored in a file format that is not natively accessible to Pig, you can optionally add the USING function to the LOAD statement to specify a user-defined function that can read in and interpret the data[16].
TRANSFORM : The transformation logic is where all the data manipulation happens. Here you can FILTER out rows that are not of interest, JOIN two sets of data files, GROUP data to build aggregations, ORDER results, and much more.
DUMP and STORE : If you don’t specify the DUMP or STORE command, the results of a Pig program are not generated. You would typically use the DUMP command, which sends the output to the screen, when you are debugging your Pig programs. When you go into production, you simply change the DUMP call to a STORE call so that any results from running your programs are stored in a file for further processing or analysis. Note that you can use the DUMP command anywhere in your program to dump intermediate result sets to the screen, which is very useful for debugging purposes.

V. HIVE

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. While initially developed by Facebook, Apache Hive is now used and developed by other companies such as Netflix. Amazon maintains a software fork of Apache Hive that is included in Amazon Elastic MapReduce on Amazon Web Services fig.3.2.
Apache Hive supports analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 filesystem[14]. It provides an SQL-like language called HiveQL with schema on read and transparently converts queries to map/reduce, Apache Tez and in the future Spark jobs. All three execution engines can run in Hadoop YARN. To accelerate queries, it provides indexes, including bitmap indexes. By default, Hive stores metadata in an embedded Apache Derby database, and other client/server databases like MySQL can optionally be used[14]. Currently, there are four file formats supported in Hive, which are TEXTFILE SEQUENCEFILE, ORC and RCFILE.
Other features of Hive include:

- Indexing to provide acceleration, index type including compaction and Bitmap index as of 0.10, more index types are planned.
- Different storage types such as plain text, RCFile, HBase, ORC, and others.

- Metadata storage in an RDBMS, significantly reducing the time to perform semantic checks during query execution.
- Operating on compressed data stored into Hadoop ecosystem, algorithm including gzip, bzip2, snappy, etc.
- Built-in user defined functions (UDFs) to manipulate dates, strings, and other data-mining tools. Hive supports extending the UDF set to handle use-cases not supported by built-in functions.
- SQL-like queries (HiveQL), which are implicitly converted into MapReduce jobs.

While based on SQL, HiveQL does not strictly follow the full SQL-92 standard. HiveQL offers extensions not in SQL, including multitable inserts and create table as select, but only offers basic support for indexes[7]. Also, HiveQL lacks support for transactions and materialized views, and only limited subquery support. There are plans for adding support for insert, update, and delete with full ACID functionality.

Internally, a compiler translates HiveQL statements into a directed acyclic graph of MapReduce jobs, which are submitted to Hadoop for execution.

Although Pig can be quite a powerful and simple language to use, the downside is that it's something new to learn and master. Some folks at Facebook developed a runtime Hadoop® support structure that allows anyone who is already fluent with SQL (which is commonplace for relational data-base developers) to leverage the Hadoop platform right out of the gate.Their creation, called Hive, allows SQL developers to write Hive Query Language (HQL) statements that are similar to standard SQL statements; now you should be aware that HQL is limited in the commands it understands, but it is still pretty useful. HQL statements are broken down by the Hive service into MapReduce jobs and executed acros a Hadoop cluster.

For anyone with a SQL or relational database background, this section will look very familiar to you. As with any database management system (DBMS), you can run your Hive queries in many ways. You can run them from a command line interface (known as the Hive shell), from a Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC) application leveraging the Hive JDBC/ODBC drivers, or from what is called a Hive Thrift Client. The Hive Thrift Client is much like any database client that gets installed on a user's client machine (or in a middle tier of a three-tier architecture): it communicates with the Hive services running on the server. You can use the Hive Thrift Client within applications written in C++, Java, PHP, Python, or Ruby (much like you can use these client-side languages with embedded SQL to access a database such as DB2 or Informix).

Hive looks very much like traditional database code with SQL access. However, because Hive is based on Hadoop and MapReduce operations, there are several key differences. The first is that Hadoop is intended for long sequential scans, and because Hive is based on Hadoop, you can expect queries to have a very high latency (many minutes). This means that Hive would not be appropriate for applications that need very fast response times, as you would expect with a database such as DB2. Finally, Hive is read-based and therefore not appropriate for transaction processing that typically involves a high percentage of write operations.

VI. HBASE

Apache HBase began as a project by the company Powerset out of a need to process massive amounts of data for the purposes of natural language search. It is now a top-level Apache project fig.3.2.Facebook elected to implement its new messaging platform using HBase in November 2010. HBase is a column-oriented database management system that runs on top of HDFS. It is well suited for sparse data sets, which are common in many big data use cases. Unlike relational database systems, HBase does not support a structured query language like SQL; in fact, HBase isn't a relational data store at all. HBase applications are written in Java much like a typical MapReduce application. HBase does support writing applications in Avro, REST, and Thrift.

An HBase system comprises a set of tables. Each table contains rows and columns, much like a traditional database. Each table must have an element defined as a Primary Key, and all access attempts to HBase tables must use this Primary Key. An HBase column represents an attribute of an object[15]; for example, if the table is storing diagnostic logs from servers in your environment, where each row might be a log record, a typical column in such a table would be the timestamp of when the log record was written, or perhaps the server name where the record originated. In fact, HBase allows for many attributes to be grouped together into what are known as column families, such that the elements of a column family are all stored together. This is different from a row-oriented relational database, where all the columns of a given row are stored together. With HBase you must predefine the table schema and specify the column families. However, it's very flexible in that new columns can be added to families at any time, making the schema flexible and therefore able to adapt to changing application requirements.

Just as HDFS has a NameNode and slave nodes, and MapReduce has JobTracker and TaskTracker slaves, HBase is built on similar concepts. In HBase a master node manages the cluster and region servers store portions of the tables and perform the work on the data[15]. In the same way HDFS has some enterprise concerns due to the availability of the NameNode (among other areas that can be "hardened" for true enterprise deployments by InfoSphere BigInsights), HBase is also sensitive to the loss of its master node.

VII. CONCLUSION

Hadoop MapReduce is a large scale, open source software framework dedicated to scalable, distributed, data-intensive computing. The framework breaks up large data into smaller parallelizable chunks and handles scheduling

▫ Maps each piece to an intermediate value
▫ Reduces intermediate values to a solution
▫ User-specified partition and combiner options

• Fault tolerant, reliable, and supports thousands of nodes and petabytes of data
• If you can rewrite algorithms into Maps and Reduces, and your problem can be broken up into small pieces solvable in parallel, then Hadoop's MapReduce is the way to go for a distributed problem solving approach to large datasets
• Tried and tested in production
• Many implementation options

We can present the design and evaluation of a data aware cache framework that requires minimum change to the original MapReduce programming model for provisioning incremental processing for Big data applications using the MapReduce model.

FUTURE ENHANCEMENT:

Usually it is observed that the MapReduce framework generates a large amount of intermediate data. Such abundant information is thrown away after the tasks finish, because MapReduce is unable to utilize them.Therefore, we propose Dache, a data-aware cache framework for big-data applications then its tasks submit their intermediate results to the cache manager. The task queries the cache manager before executing the actual computing work. A novel cache description scheme and a cache request and reply protocol are designed.

REFERENCES

[1] Dhole Poonam B, Gunjal Baisa L, "*Survey Paper on Traditional Hadoop and Pipelined Map Reduce*" International Journal of Computational Engineering Research||Vol, 03||Issue, 12||

[2] Nilam Kadale, U. A. Mande, "*Survey of Task Scheduling Method for Mapreduce Framework in Hadoop*" International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA 2nd National Conference on Innovative Paradigms in Engineering & Technology (NCIPET 2013) – *www.ijais.org*

[3] Suman Arora, Dr.Madhu Goel, "*Survey Paper on Scheduling in Hadoop*" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 5, May 2014

[4] Wang, F. *et al*. Hadoop High Availability through Metadata Replication. *ACM* (2009).

[5] B.Thirumala Rao, Dr. L.S.S.Reddy, "*Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments*", *International Journal of Computer Applications (0975 – 8887) Volume 34– No.9, November 2011*

[6] Amogh Pramod Kulkarni, Mahesh Khandewal, "*Survey on Hadoop and Introduction to YARN*", International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 5, May 2014)

[7] Vishal S Patil, Pravin D. Soni**, "***HADOOP SKELETON & FAULT TOLERANCE IN HADOOP CLUSTERS***",** International Journal of Application or Innovation in Engineering & Management (IJAIEM)Volume 2, Issue 2, February 2013 ISSN 2319 - 4847

[8] Sanjay Rathe, "*Big Data and Hadoop with components like Flume, Pig, Hive and Jaql*" International Conference on Cloud, Big Data and Trust 2013, Nov 13-15, RGPV

[9] Yaxiong Zhao, Jie Wu and Cong Liu, "*Dache: A Data Aware Caching for Big-Data Applications Using the MapReduce Framework*",TSINGHUA SCIENCE AND TECHNOLOGY ISSN 1007-0214 05/10 pp39-50 Volume 19, Number 1, February 2014

[10] Parmeshwari P. Sabnis, Chaitali A.Laulkar , "*SURVEY OF MAPREDUCE OPTIMIZATION METHODS*", ISSN (Print): 2319-2526, Volume -3, Issue -1, 2014

[11] Puneet Singh Duggal ,Sanchita Paul ," *Big Data Analysis: Challenges and Solutions*", *International Conference on Cloud, Big Data and Trust 2013, Nov 13-15, RGPV*

[12] Chen He,Ying Lu,David Swanson, "*Matchmaking: A New MapReduce Scheduling Technique*", EECS Department, University of California, Berkeley, Tech. Rep.,April 2009

[13] Apache HDFS. Available at http://hadoop.apache.org/hdfs

[14] Apache Hive. Available at http://hive.apache.org

[15] Apache HBase. Available at http://hbase.apache.org

[16] Apache Pig. Available at http://pig.apache.org