Survey on Big Data Technologies

Recent technologies on big data

A. Pratheeba,
M.Tech II yr,
K.L.N.College of Information Technology,
Pottapalayam, Sivagangai (dist),

S. K. Karthikumar,
Associate professor , IT Department,
K.L.N.College of Information Technology,
Pottapalayam, Sivagangai (dist),

Dr. S. Appavu Alias Balamurugan,
Head of the IT Department,
K.L.N.College of Information Technology, Pottapalayam,Sivagangai (dist),

***Abstract*—In recent years handling of large data becomes more complex so the processing demand will be high on big data. In worldwide there are 9 geographic regions are used to store the data. Analyzing of big data is a challenging task to handle this situation Hadoop Distributed File System is used. This paper provides the overview of big data and hadoop architecture ,challenges of big data, tools used on big data and various methods to solve the problems in hand through Map Reduce Framework over Hadoop distributed File System. Map Reduce is a minimization technique which makes use of indexing with mapping, sorting, shuffling and finally reducing. implementation of big data combines both infrastructure and analytics. When analysed the big data properly, it provides new business insights, open new markets, and create competitive advantage.**

Keywords*— *Big data,HDFS,Map reduce

I. INTRODUCTION

The ability to manage large datasets is becoming more important in research and business environments. Big data is the term for data sets so large and complicated that it becomes difficult to process using traditional data management tools or processing applications. Big Data is a heterogeneous mix of data both structured (traditional datasets –in rows and columns like DBMS tables, CSV's and XLS's) and unstructured data like e-mail attachments, manuals, images, PDF documents, medical records such as x-rays, ECG and MRI images, forms, rich media like graphics, video and audio, contacts, forms and documents.

Defining Big Data

Big data typically refers to the following types of data:

a) Traditional enterprise data – includes customer information from CRM systems, transactional ERP data, web store transactions, and general ledger data.
b) Machine-generated /sensor data – includes Call Detail Records ("CDR"), weblogs, smart meters, manufacturing sensors, equipment logs (often referred to as digital exhaust), trading systems data.
c) Social data – includes customer feedback streams, micro-blogging sites like Twitter, social media platforms like Facebook

The McKinsey Global Institute estimates that data volume is growing 40% per year, and will grow 44x between 2009 and 2020. But while it's often the most visible parameter, volume of data is not the only characteristic that matters. In fact, there are four key characteristics that define big data:

a) Volume.

Machine-generated data is produced in much larger quantities than non-traditional data. For instance, a single jet engine can generate 10TB of data in 30 minutes. With more than 25,000 airline flights per day, the daily volume of just this single data source runs into the Petabytes. Smart meters and heavy industrial equipment like oil refineries and drilling rigs generate similar data volumes, compounding the problem.

b) Velocity.

Social media data streams – while not as massive as machine-generated data – produce a large influx of opinions and relationships valuable to customer relationship management. Even at 140 characters per tweet, the high velocity (or frequency) of Twitter data ensures large volumes (over 8 TB per day).

c) Variety.

Traditional data formats tend to be relatively well defined by a data schema and change slowly. In contrast, non-traditional data formats exhibit a dizzying rate of change. As new services are added, new sensors deployed, or new marketing campaigns executed, new data types are needed to capture the resultant information.

d) Value.

The economic value of different data varies significantly. Typically there is good information hidden amongst a larger body of non-traditional data; the challenge is identifying what is valuable and then transforming and extracting that data for analysis.

To make the most of big data, enterprises must evolve their IT infrastructures to handle these new high-volume, high-velocity, high-variety sources of data and integrate them with the pre-existing enterprise data to be analyzed.

The Importance of Big Data

When big data is distilled and analyzed in combination with traditional enterprise data, enterprises can develop a more thorough and insightful understanding of their business, which can lead to enhanced productivity, a stronger competitive position and greater innovation – all of which can have a significant impact on the bottom line.

For example, in the delivery of healthcare services, management of chronic or long-term conditions is expensive. Use of in-home monitoring devices to measure vital signs, and monitor progress is just one way that sensor data can be used to improve patient health and reduce both office visits and hospital admittance.

Manufacturing companies deploy sensors in their products to return a stream of telemetry. In the automotive industry, systems such as General Motors' OnStar ® or Renault's R-Link ®, deliver communications, security and navigation services. Perhaps more importantly, this telemetry also reveals usage patterns, failure rates and other opportunities for product improvement that can reduce development and assembly costs.

The proliferation of smart phones and other GPS devices offers advertisers an opportunity to target consumers when they are in close proximity to a store, a coffee shop or a restaurant. This opens up new revenue for service providers and offers many businesses a chance to target new customers. Retailers usually know who buys their products. Use of social media and web log files from their ecommerce sites can help them understand who didn't buy and why they chose not to, information not available to them today. This can enable much more effective micro customer segmentation and targeted marketing campaigns, as well as improve supply chain efficiencies through more accurate demand planning.

Finally, social media sites like Facebook and LinkedIn simply wouldn't exist without big data. Their business model requires a personalized experience on the web, which can only be delivered by capturing and using all the available data about a user or member.

II HADOOP AND HDFS

Hadoop is a scalable, open source, fault-tolerant Virtual Grid operating system architecture for data storage and processing. It runs on commodity hardware, it uses HDFS which is fault-tolerant high-bandwidth clustered storage architecture. It runs MapReduce for distributed data processing and is works with structured and unstructured data.

Figure1Illustrates the layers found in the software architecture of aHadoop stack. At the bottom of the Hadoop software stack is HDFS, a distributed file system in which each file appears as a (very large) contiguous and randomly addressable sequence of bytes. For batch analytics, the middle layer of the stack is the Hadoop Map Reduce system, which applies map operations to the data in partitions of an HDFS file, sorts and redistributes the results based on key values in the output data, and then performs reduce operations on the groups of output data items with matching keys from the map phase of the job. For applications just needing basic key-based record management operations, the HBase store (layered on top of HDFS) is available as a key-value layer in the Hadoop stack. As indicated in the figure, the contents of HBase can either be directly accessed and manipulated by a client application or accessed via Hadoop for analytical needs. Many users of the Hadoop stack prefer the use of a declarative language over the bare MapReduce programming model. High-level language compilers (Pig and Hive) are thus the topmost layer in the Hadoop software stack for such clients.

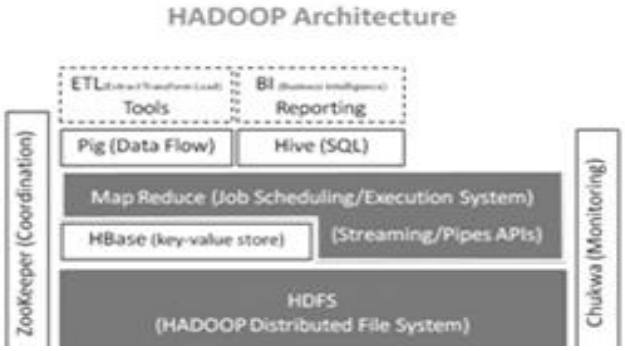

Figure 1.Hadoop Architecture Layers

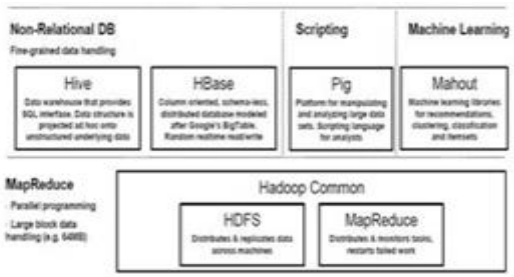

Figure 2.Hadoop Architecture Tools and usage

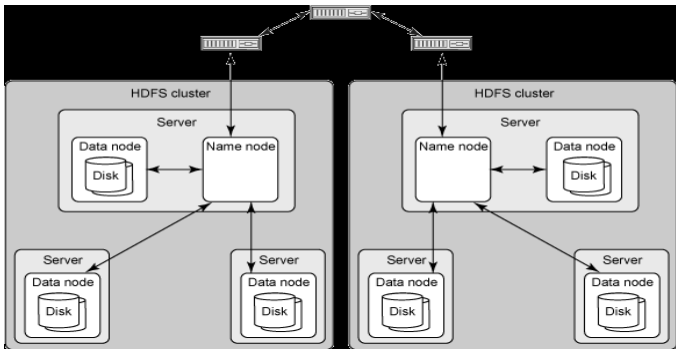

Figure 3. HDFS Clusters

Figure2 shows the relevancy between the traditional experience in data warehousing, reporting, and online analytic processing (OLAP) and advanced analytics with collection of related techniques like data mining with DBMS, artificial intelligence, machine learning, and database analytics platforms such as MapReduce and Hadoop over HDFS.

Figure3 shows the architecture of HDFS clusters implementation with Hadoop. It can be seen that HDFS has distributed the task over two parallel clusters with one server and two slave nodes each. Data analysis tasks are distributed in these clusters.

III TECHNOLOGIES

a) Column-oriented databases

Traditional, row-oriented databases are excellent for online transaction processing with high update speeds, but they fall short on query performance as the data volumes grow and as data becomes more unstructured. Column-oriented databases store data with a focus on columns, instead of rows, allowing for huge data compression and very fast query times. The downside to these databases is that they will generally only allow batch updates, having a much slower update time than traditional models.

b) Schema-less databases, or NoSQL databases

There are several database types that fit into this category, such as key-value stores and document stores, which focus on the storage and retrieval of large volumes of unstructured, semi-structured, or even structured data. They achieve performance gains by doing away with some (or all) of the restrictions traditionally associated with conventional databases, such as read-write consistency, in exchange for scalability and distributed processing.

c) MapReduce

This is a programming paradigm that allows for massive job execution scalability against thousands of servers or clusters of servers. Any MapReduce implementation consists of two tasks:

The "Map" task, where an input dataset is converted into a different set of key/value pairs, or tuples;

The "Reduce" task, where several of the outputs of the "Map" task are combined to form a reduced set of tuples (hence the name).

d) Hadoop

Hadoop is by far the most popular implementation of MapReduce, being an entirely open source platform for handling Big Data. It is flexible enough to be able to work with multiple data sources, either aggregating multiple sources of data in order to do large scale processing, or even reading data from a database in order to run processor-intensive machine learning jobs. It has several different applications, but one of the top use cases is for large volumes of constantly changing data, such as location-based data from weather or traffic sensors, web-based or social media data, or machine-to-machine transactional data.

e) Hive

Hive is a "SQL-like" bridge that allows conventional BI applications to run queries against a Hadoop cluster. It was developed originally by Facebook, but has been made open source for some time now, and it's a higher-level abstraction of the Hadoop framework that allows anyone to make queries against data stored in a Hadoop cluster just as if they were manipulating a conventional data store. It amplifies the reach of Hadoop, making it more familiar for BI users.

f) PIG

PIG is another bridge that tries to bring Hadoop closer to the realities of developers and business users, similar to Hive. Unlike Hive, however, PIG consists of a "Perl-like" language that allows for query execution over data stored on a Hadoop cluster, instead of a "SQL-like" language. PIG was developed by Yahoo!, and, just like Hive, has also been made fully open source.

g) WibiData

WibiData is a combination of web analytics with Hadoop, being built on top of HBase, which is itself a database layer on top of Hadoop. It allows web sites to better explore and work with their user data, enabling real-time responses to user behavior, such as serving personalized content, recommendations and decisions.

h) PLATFORA

Perhaps the greatest limitation of Hadoop is that it is a very low-level implementation of MapReduce, requiring extensive developer knowledge to operate. Between preparing, testing and running jobs, a full cycle can take hours, eliminating the interactivity that users enjoyed with conventional databases. PLATFORA is a platform that turns user's queries into Hadoop jobs automatically, thus creating an abstraction layer that anyone can exploit to simplify and organize datasets stored in Hadoop.

i) Storage Technologies

As the data volumes grow, so does the need for efficient and effective storage techniques. The main evolutions in this space are related to data compression and storage virtualization.

j) SkyTree

SkyTree is a high-performance machine learning and data analytics platform focused specifically on handling Big Data. Machine learning, in turn, is an essential part of Big Data, since the massive data volumes make manual exploration, or even conventional automated exploration methods unfeasible or too expensive.

IV MAP REDUCE

MapReduce is a programming model for processing large-scale datasets in computer clusters. The MapReduce programming model consists of two functions, map() and reduce(). Users can implement their own processing logic by

specifying a customized map() and reduce() function. The map() function takes an input key/value pair and produces a list of intermediate key/value pairs. The MapReduce runtime system groups together all intermediate pairs based on the intermediate keys and passes them to reduce() function for producing the final results.

Map (in_key, in_value) --->list(out_key,intermediate_value)
Reduce (out_key,list(intermediate_value)) -- ->list(out_value)

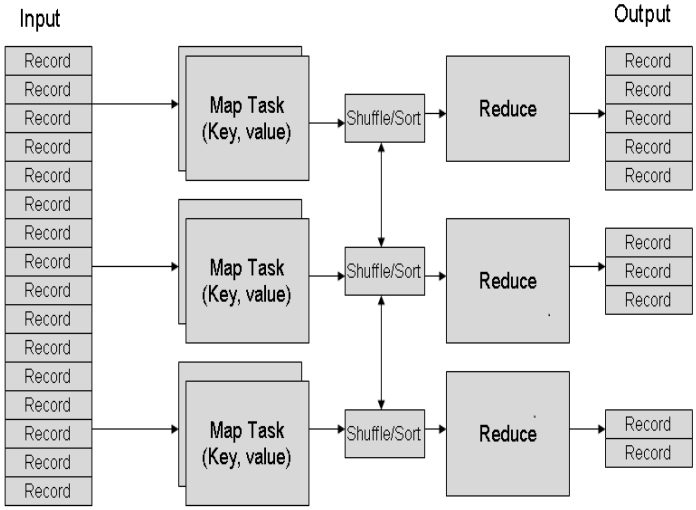

Figure 4. Map Reduce Architecture and Working

The signatures of map() and reduce() are as follows :

map (k1,v1) ! list(k2,v2)and reduce (k2,list(v2)) ! list(v2)

A MapReduce cluster employs a master-slave architecture where one master node manages a number of slave nodes . In the Hadoop, the master node is called JobTracker and the slave node is called TaskTracker as shown in the figure 4. Hadoop launches a MapReduce job by first splitting the input dataset into even-sized data blocks. Each data block is then scheduled to one TaskTracker node and is processed by a map task. The TaskTracker node notifies the JobTracker when it is idle. The scheduler then assigns new tasks to it. The scheduler takes data locality into account when it disseminates data blocks.

It always tries to assign a local data block to a TaskTracker. If the attempt fails, the scheduler will assign a rack-local or random data block to the TaskTracker instead. When map() functions complete, the runtime system groups all intermediate pairs and launches a set of reduce tasks to produce the final results. Large scale data processing is a difficult task, managing hundreds or thousands of processors and managing parallelization and distributed environments makes is more difficult. Map Reduce provides solution to the mentioned issues, as is supports distributed and parallel I/O scheduling, it is fault tolerant and supports scalability and it has inbuilt processes for status and monitoring of data and large datasets as in Big Data.

A. Map Reduce Components

1. Name Node – manages HDFS metadata, doesn‟t deal with files directly

2. Data Node – stores blocks of HDFS – default replication level for each block: 3

3. Job Tracker – schedules, allocates and monitors job execution on slaves – Task Trackers

4. Task Tracker – runs Map Reduce operations

B. Map Reduce Working

We implement the Mapper and Reducer interfaces to provide the map and reduce methods as shown in figure 4. These form the core of the job.

1) Mapper

Mapper maps input key/value pairs to a set of intermediate key/value pairs. Maps are the individual tasks that transform input records into intermediate records. The transformed intermediate records do not need to be of the same type as the input records. A given input pair may map to zero or many output pairs.

The number of maps is usually driven by the total size of the inputs, that is, the total number of blocks of the input files. The right level of parallelism for maps seems to be around 10-100 maps per-node, although it has been set up to 300 maps for very cpu-light map tasks. Task setup takes awhile, so it is best if the maps take at least a minute to execute. For Example, if you expect 10TB of input data and have a blocksize of 128MB, you'll end up with 82,000 maps.

2) Reducer

Reducer reduces a set of intermediate values which share a key to a smaller set of values.
Reducer has 3 primary phases: shuffle, sort and reduce.

2.1) Shuffle

Input to the Reducer is the sorted output of the mappers. In this phase the framework fetches the relevant partition of the output of all the mappers, via HTTP.
Name Node–manages HDFS metadata, doesn't deal with files directly.
Data Node –stores blocks of HDFS –default replication level for each block: 3
Job Tracker –schedules, allocates and monitors job execution on slaves –Task Trackers
Task Tracker –runs Map Reduce operations

2.2) Sort

The framework groups Reducer inputs by keys (since different mappers may have output the same key) in this stage. The shuffle and sort phases occur simultaneously; while map-outputs are being fetched they are merged.

2.3) Secondary Sort

If equivalence rules for grouping the intermediate keys

are required to be different from those for grouping keys before reduction, then one may specify a Comparator (Secondary Sort).

2.4) Reduce

In this phase the reduce method is called for each <key, (list of values)> pair in the grouped inputs.The output of the reduce task is typically written to the File System via Output Collector.

Applications can use the Reporter to report progress, set application-level status messages and update Counters, or just indicate that they are alive. The output of the Reducer is *not sorted*. The right number of reduces seems to be 0.95 or 1.75 multiplied by *no. of nodes.* With 0.95 all of the reduces can launch immediately and start transferring map outputs as the maps finish. With 1.75 the faster nodes will finish their first round of reduces and launch a second wave of reduces doing a much better job of load balancing [MR Framework].Increasing the number of reduces increases the framework overhead, but increases load balancing and lowers the cost of failures. The scaling factors above are slightly less than whole numbers to reserve a few reduce slots in the framework for speculative-tasks and failed tasks. It is legal to set the number of reduce-tasks to *zero* if no reduction is desired.

a) Partitioner

Partitioner partitions the key space. Partitioner controls the partitioning of the keys of the intermediate map-outputs. The key (or a subset of the key) is used to derive the partition, typically by a *hash function*. The total number of partitions is the same as the number of reduce tasks for the job. Hence this controls which of the m reduce tasks the intermediate key (and hence the record) is sent to for reduction.
Hash Partitioner is the default Partitioner.

b) Reporter

Reporter is a facility for MapReduce applications to report progress, set application-level status messages and update Counters. Mapper and Reducer implementations can use the Reporter to report progress or just indicate that they are alive. In scenarios where the application takes a significant amount of time to process individual key/value pairs, this is crucial since the framework might assume that the task has timed-out and kill that task. Applications can also update Counters using the Reporter.

c) Output Collector

Output Collector is a generalization of the facility provided by the MapReduce framework to collect data output by the Mapper or the Reducer (either the intermediate outputs or the output of the job). HadoopMapReduce comes bundled with a library of generally useful mappers, reducers, and partitioners. Map Reduce Working through Master / Slave.

C. Map Reduce techniques

Combining

Combiners provide a general mechanism within the MapReduce framework to reduce the amount of intermediate data generated by the mappers. They can be understood as "mini-reducers" that process the output of mappers. The combiner's aggregate term counts across the documents processed by each map task. This result in a reduction in the number of intermediate key-value pairs that need to be shuffled across the network, from the order of total number of terms in the collection to the order of the number of unique terms in the collection. They reduce the result size of map functions and perform reduce-like function in each machine which decreases the shuffling cost.

Inverse Indexing

Inverse indexing is a technique in which the keywords of the documents are mapped according to the document keys in which they are residing.

For example Doc1: IMF, Financial Economics Crisis Doc2: IMF, Financial Crisis Doc3: Harry Economics Doc4: Financial Harry Potter Film Doc5: Harry Potter Crisis The following is the inverted index of the above data IMF -> Doc1:1, Doc2:1 Financial -> Doc1:6, Doc2:6, Doc4:1 Economics -> Doc1:16, Doc3:7 Crisis -> Doc1:26, Doc2:16, Doc5:14 Harry -> Doc3:1, Doc4:11, Doc5:1 Potter -> Doc4:17, Doc5:7 Film -> Doc4:24

Shuffling

Shuffling is the procedure of mixing the indexes of the files and their keys, so that a heterogeneous mix of dataset can be obtained. If the dataset is shuffled, then there are better chances that the resultant query processing will yield near accurate results. We can relate the shuffling process with the population generating by crossover in the GA algorithms. The processes are different in nature, but their purpose is similar.

Sharding

It is a term used to distribute the Mappers in the HDFS architecture. Sharding refers to the groupings or documents which are done so that the MapReduce jobs are done parallel in a distributed environment.

Joins

Join is a RDBMS term; it refers to combining two or more discrete datasets to get Cartesian product of data of all the possible combinations. Map Reduce does not have its own Join techniques, but RDBMS techniques are tweaked and used to get the maximum possible combinations. The join techniques which are adopted for Map Reduce are Equi Join, Self Join, Repartition Join and Theta Join.

Clustering **&** Classification

Data Analysis term, used mainly in Data Mining.

In Map Reduce it is achieved through K means clustering. Here, iterative working improves partitioning of data into k clusters. After the clustering, the data sorted are grouped together based upon rules to be formed into classes. The steps for clustering in Map Reduce are; Step1: Do Step2: Map Step3: Input is a data point and k centres are broadcasted

Step4: Finds the closest centre among k centres for the input point Step5: Reduce Step6: Input is one of k centres and all data points having this centre as their closest centre Step7: Calculates the new centre using data points Step 8: Repeat 1-7, until all of new centres are not changed

V CONCLUSION

Processing of enormous quantities of data becomes more complex. In this paper we present the detailed view about the big data ,Hadoop architecture ,map reduce framework and challenges of big data. By using Big Data analysis tools like Map Reduce over Hadoop and HDFS, promises to help organizations better understand their customers and the marketplace, hopefully leading to better business decisions and competitive advantages.

REFERENCES

[1] Steve Loughran, Jose M. Alcaraz Calero,Andrew Farrell,Johannes Kirschnick, and Julio Guijarro Hewlett-Packard Laboratories" Dynamic Cloud Deployment of a MapReduce Architecture".

[2] "Data from Year 2000 US Census," http://aws.amazon.com/dataset s/Economics/2290.

[3] "Department of Defense Information Enterprise Strategic Plan 2011- 2012," http://dodcio.defense.gov/docs/DodIESP-r16.pdf.

[4] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen, "Don't Settle for Eventual: Scalable Causal Consistency for Wide- area Storage with COPS," in SOSP, 2011.

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

[6] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in OSDI, 2004.

[7] Apache Software Foundation, "Hadoop," http://hadoop.apache.org.

[8] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated Data Placement for Geo-Distributed Cloud Services," in NSDI, 2010.