ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

A Survey on Big Data Analytics and MapReduce Operations on Distributed Systems

Ajinkya Kunjir, Basil Shaikh

UG Student, Dept. of Computer Engineering, M.E.S College of Engineering, Pune, India

UG Student, Dept. of Computer Engineering, M.E.S College of Engineering, Pune, India

ABSTRACT: MapReduce is a programming model used for generating and processing large datasets and terabytes of data across multiple clusters. There are two functions of this model 'Map' and 'Reduce'.'Map' function is used to generate intermediate key and value pair, and 'Reduce' function is used to merge all the key-value pairs generated.Programs and snippet written using this framework model can be executed on parallel and distributed platforms. The characteristics of this model such as analysis and fault tolerance allows programmers to implement their programs on distributed and parallel programs. This paper outlines the operations of MapReducemodel and big data analytics on the distributed systems. Big data is any data that has potential to be mined. Most of the data is unstructured and we need a way to manage this data or rather generate important information. MapReduce was invented by Google in 2004 for running applications across massive datasets, on huge clusters of machines comprising of commodity hardware capable of processing terabytes of data. It implements this computational paradigm used in functional programming. In simple terms its a divide and conquer technique where the origin data is divided into self contained units of work, each unit is executed independently on any node in the cluster, a key to Map/Reduce programming model.

KEYWORDS*:* Big data ,Information retrieval , Data visualization, MapReduce, data analysis, Distributed databases**.**

I. INTRODUCTION

Over the past five years, the developers and researchers at Google have implemented thousands of special-purpose computations that process huge amounts of raw data, such as formatted documents, web request, server logs, etc., to compute various kinds of derived data, such as various representations of the graphical structure of web documents, summaries of the number of pages crawled per host, the set of most frequent queries in a day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computationshave to be distributed across hundreds or thousands of machines in order to finish in a feasible amount of time. The issues of how to distribute and parallelize the computation, distribute the data, and handle failures conspire to run the original easy computation with large amounts of complex code to deal with these issues.

Jefferey Dean and Sanjay Ghemawat in their paper of 'Google Inc' stated tha tthe reaction to this complexity, there was designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the noisy details of parallelization, fault-tolerance, data distribution and load balancing in a library[1]. The abstraction was inspired by the map and reduce primitives present in Lisp and many other functional languages.] Howard Karlo et.al in their paper derived a theory, that most of the computations involved applying a map operation to each logical "record" in our input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key, in order to combine the derived data appropriately[2]. The utilization of a functional model with user specified map and reduce operations allows us to parallelize large computations easily and to use re-execution as the primary mechanism for fault tolerance. The major contributionsof this work were a simple and powerful linterface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves highper for mance on large clusters of commodity PCs.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

Section2describesthe previous researches, basic program mingmode lsand gives several examples. Section 3 describes the operations of MapReduce, and the architecture of MapReduce systems. Section 4 explains about big data and analysis use to handle big data in real time and business use. Section 5discusses related and futurework.

II. RELATED WORK

A. BACKGROUND AND RESEARCH

The MapReduce framework was originally developed at Google , but has recently seen wide adoption across all the platforms and has become the usual standard for large scale data analysis. Publicly available statistics indicate that MapReduce is used to process more than 10 petabytes of information per day at Google alone. The Wikipedia source from the internet provided information about anopensourceversion, calledHadoop, which is also a data handling framework hasrecently been developed, and is seeing increased adoption both in industry and academia[3]. Over 80 companies use Hadoop including Yahoo!, New York Stock Market, Facebook, Flash, and IBM. Moreover, Amazon's Elastic Compute Cloud(EC2)isaHadoopclusterwhereuserscanupload large data sets and rent processor time[4]. In addition, at least seven universities (including CMU, Cornell, and the University of Maryland) are using Hadoop clusters for research. Hadoop and MapReduce together can solve various issues of computing across distributed and parallel platforms.MapReduce can also be implemented using efficient scheduling schemes and also operations across all the nodes in the distributed systems.

B. PROGRAMMING MODELS: A SURVEY

The computation of MapReduce takes a plethora of input key/value pairs, and produces a plethora of output key/value pairs. The accesser of the MapReduce library expresses the computation as two functions: Map and Reduce. Map, a user defined function takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key and passes them to the Reduce function. The Reduce function, also written by the user, accepts an intermediate key and a set of values for that key. It combines together these values to form a possibly smaller set of values. Ideally, just zero or one output value is produced per Reduce iteartion. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory. Also this model is efficient and usable. usable.

- *Mappers and Reducers:*

Key-value pairs constitute the basic data structure in MapReduce. J. Feldmanet.al in their edition of 2008 gave a brief description about the keys and values, where it was stated that Keys and values may be primitives such as integers, floating point values, double units, strings, and raw bytes, or they may be arbitrarily complex structures likelists, tuples, associative arrays, etc[5]. Programmers typically need to define their own custom data types, although a number of libraries such as Protocol Bu□ers, Thrift, and Avro simplify the task. Part ofthedesignofMapReduce algorithmsinvolvesimposingthekey-valuestructure on arbitrary datasets. For a set of web pages, keys may be URLs and values may be the actual HTML content. For a graph, keys may represent node identifiers(id's) and values may contain the adjacency lists of those nodes. In few algorithms, input keys are not particularly meaningful and are simply ignored during processing, while in other cases input keys are used to uniquely identify a datum (such as a record id). Previously, we discussed the role of complex keys and values in the design of various algorithms. In MapReduce, the programmer defines a mapper and a reducer with the following signatures:

$$\text{map: } (k1,v1) \rightarrow [(k2,v2)] \quad \text{eq.(1)}$$

$$\text{reduce: } (k2,[v2]) \rightarrow [(k3,v3)] \quad \text{eq.(2)}$$

As we can see from equation (1) and (2) which are the conventions for map and reduce which also states that the input to a MapReduce job starts as data stored on the underlying distributed file system. The mapper is applied to every input key-value pair to generate an arbitrary number of intermediate key-value pairs. The reducer is applied to all values associated with the same intermediate key to generate output key-value pairs. Implicit between the map and reduce phases is a distributed "group by" operation on intermediate keys. Intermediate data arrive at each reducer in order, sorted by the key. However, no ordering relationship is guaranteed for keys across di□erent reducers. Output key-value pairs from each reducer are written back onto the distributed file.Theoutputendsup in 'r' filesonthedistributedfilesystem, where 'r' is the number of reducers. For the most part, there is no need to consolidate reducer output, since the r files often serve as input to yet another MapReduce job. Figure 1 illustrates this two-stage processing structure.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

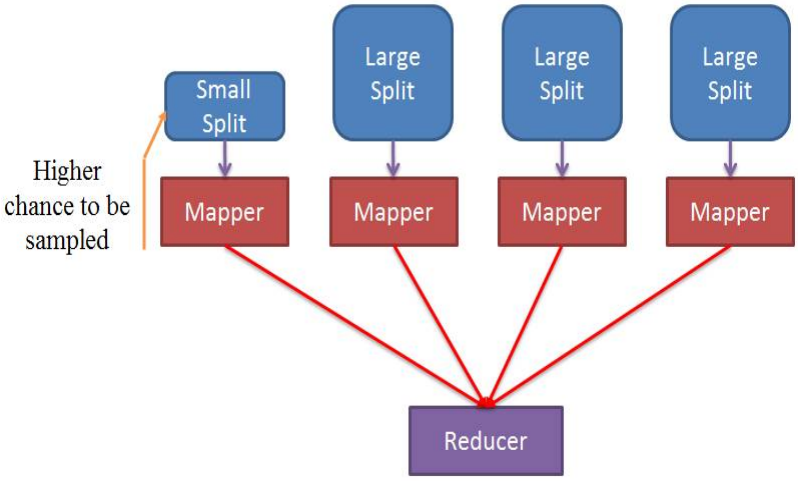

Fig 1 : Mapper and Reducer

- *Partitioners and Combiners :*

We have so far seen a simplified view of MapReduce. There are two additional components that complete the programming model:partitioners and combiners. Partitioners are responsible for dividing up the intermediate key space and assigning intermediate key-value pairs to reducers. In simple words, the partitioner specifies the task to which an intermediate key-value pair must be copied. Within each reducer, keys are processed in sorted. The simplest partitioner involves computing the hash value of the key and then taking the mod of that value with the number of reducers. This assigns approximately the same number of keys to each reducer. Note, however, that the partitioner only considers the key and ignores the value—therefore, a roughly-even partitioning of the key space may nevertheless yield large di□erences in the number of key-values pairs sent to each reducer. This imbalance in the amount of data associated with each key is relatively common in many text processing applications due to the Zipfian distribution of word occurrences. Combiners are an optimization in MapReduce that allow for local aggregation beforetheshu□eandsortphase.We can motivate the need for combinersby considering the Figure 2, which emits a key-value pair for each word in the collection. Furthermore, all these key-value pairs need to be copied across the network, and so the amount of intermediate data will be larger than the input collection itself. This is clearly ine□cient. One solution is to perform local aggregation on the output of each mapper, i.e., to compute a local count for a word over all the documents processed by the mapper. With this modification, the number of intermediate key-value pairs will be at most the number of unique words in the collection times the number of mappers. The combiner in MapReduce supports such an optimization. One can think of combiners as “mini-reducers” that take place on the output of the mappers, prior to the shu□e and sort phase. Each combiner operates in isolation and therefore does not have access to intermediate output from other mappers. The combiner is provided keys and values associated with each key. Critically, one cannot assume that a combiner will have the opportunity to process all values associated with the same key. The combiner can emit any number of key-value pairs, but the keys and values must be of the same type as the mapper output. In cases where an operation is both associative and commutative (e.g., addition or multiplication), reducers can directly serve as combiners. In general, however, reducers and combiners are not interchangeable.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

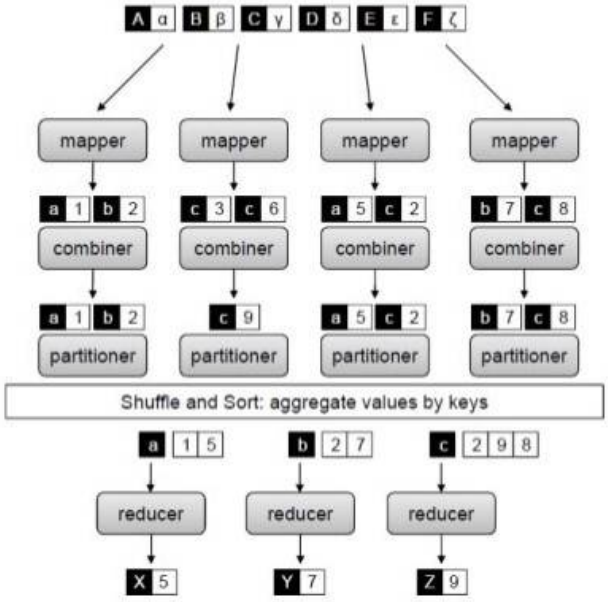

Fig 2. Partitioners and Combiners

C. MAPREDUCE COMPUTATIONS

Consider the well known problem of counting the number of occurrences of each word in a large set of documents. The user would write code similar to the following pseudo-code:

```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value:
EmitIntermediate(w, "1");
reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
result += ParseInt(v);
Emit(AsString(result));
```

The map function emits each word and an associated count of occurrences. The reduce function adds together all counts invoked for a particular word. In addition, the user writes code to fill in a Mapreduce specification object with the names of the input and output files, and optional tuning parameters. The user then invokes the MapReduce function, passing it the specification object[2]. The user's code is linked together with the MapReduce library.Here are a few simple examples of interesting programs that can be easily expressed as MapReduce computations.

1. Distributed Grep: The map function emits a line if it matches a supplied pattern. The reduce function is an identity function that just copies the supplied intermediate data to the output.
2. Count of URL Access Frequency: The map function processes logs of web page requests and outputs the URL. The reduce function combines all the values for the same URL and emits a URL, total count pair.
3. ReverseWeb-LinkGraph: Themapfunction outputs the target, source pairs for each link to a target URL found in a page named source. The reduce function concatenates the list of all source URLs associated with a given target URL and emits the pair target,list(source).
4. Term-VectorperHost: A term vectors ummarizes the most important words that occur in a document or a set of documents as a list of word-frequency pairs. The map function emits a hostname, term vector pair for each

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

input document. The reduce function is passed all per-document term vectors for a given host. It adds these term vectors together, throwing away infrequent terms, and then emits a final hostname term vector pair.

III. MAP-REDUCE OPERATIONS: A SURVEY

A. *MAPREDUCE IMPLEMENTATION:*

• Input data is partitioned into 'M' splits
• Map: extract information on each split – Each Map produces 'R' partitions
• Shuffle and sort – Bring 'M' partitions to the same reducer
• Reduce: aggregate, summarize, filter or transform
• Output is in 'R' result file

B. *MAPREDUCE SCHEDULING :*

• Master assigns a map task to a free worker
– Prefers "close-by" workers when assigning task
– Worker reads task input (From Local Disk)
– Worker produces 'R' local files containing intermediate key-value pairs
• Master assigns a reduce task to a free worker
– Worker reads intermediate k/v pairs from map workers
– Worker sorts & applies user's Reduce operation to produce the output.

Task / Job Scheduling : Scheduling in MapReduce is a concept in which the slave node send heartbeats periodically to which the master responds with task if a slot is free, picking task with dataset closest to the node. Problem for job scheduling in MapReduce is the poor locality for small jobs, which is almost 58% for the jobs of size less than 25 units. The second problem is the sticky slots, which occurs when the task slots are divided equally between the jobs . The other concerns are memory aware resource scheduling, throughput gains and network traffic reduction. A solution to all these problems is global scheduling. The further analysis for throughput is Always worth it, unless there's a hotspot.If hotspot, prefer to run IO-bound tasks on the hotspot node and CPU-bound tasks remotely (rationale: maximize rate of local input output) .

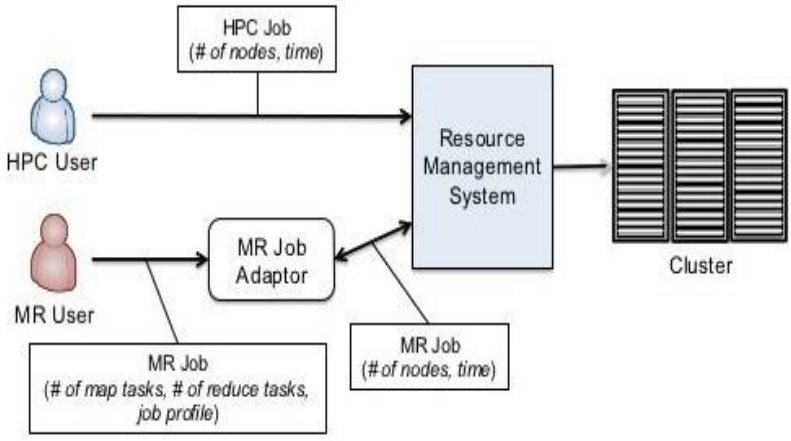

Fig 3 :MapReduce Job Scheduling

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

IV. MAP-REDUCE ARCHITECTURE

A. MAPREDUCE FOR TRADITIONAL SYSTEMS

MapReduce processes data in the form of key-value pairs. A key-value (KV) pair is a mapping element between two linked data items - key and its value. The key (K) acts as an identifier to the value. An example of a key-value (KV) pair is a pair where the key is the node Id and the value is its properties including neighbour nodes, predecessor node, etc. MR API provides the following features like batch processing, parallel processing of huge amounts of data and high availability. For processing large sets of data MR comes into the picture. The programmers will write MR applications that could be suitable for their business scenarios. Programmers have to understand the MR working flow and according to the flow, applications will be developed and deployed across Hadoop clusters. Hadoop built on Java APIs and it provides some MR APIs that is going to deal with parallel computing across nodes.The MR work flow undergoes different phases and the end result will be stored in 'HDFS' with replications. Job tracker is going to take care of all MR jobs that are running on various nodes present in the Hadoop cluster. Job tracker plays vital role in scheduling jobs and it will keep track of the entire map and reduce jobs. Actual map and reduce tasks are performed by Task tracker.

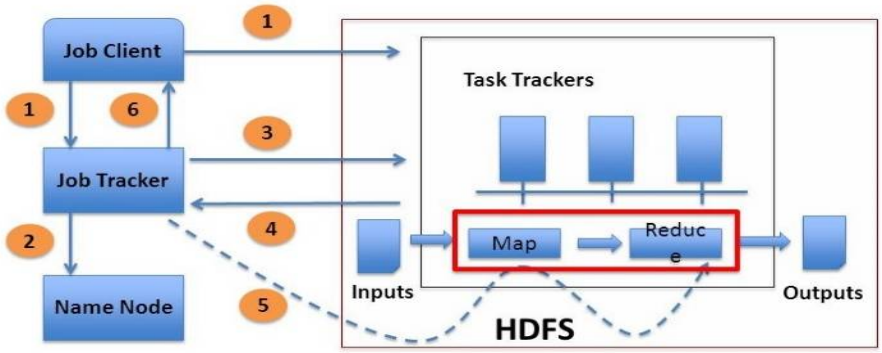

Fig 4 :Mapreduce Workflow Architecture

B. MAPREDUCE FOR PARALLEL SYSTEMS

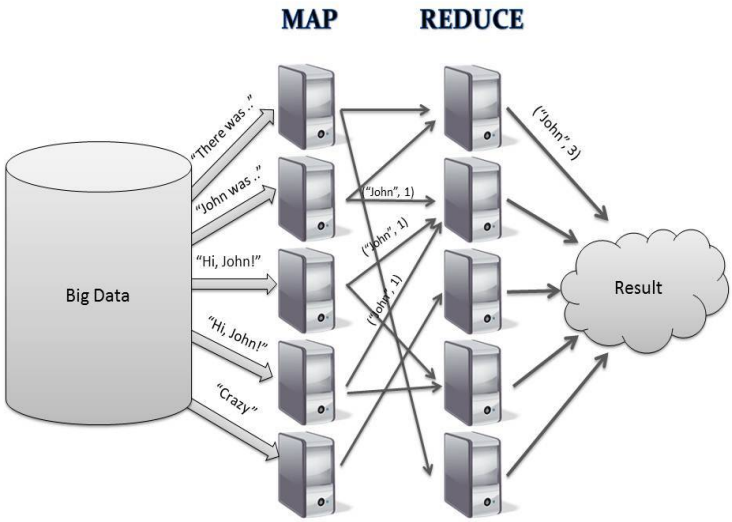

Fig 5 : Parallel MapReduce Architecture

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

D. K. G. Campbell in his survey of parallel computations elaborated the use of MapReduce by this parallel programming concept and is a mechanism for processing huge amounts of raw data[6]. For example web request logs, Website URLs etc. This data is so large; it must be distributed across thousands of machines in order to be processed in a reasonable time. L. G. Valiant in the early 90's in his paper derived that the distribution implies parallel computing since the same programs are performed on each CPU, but with a different dataset. Here dataset are not depending on each other during execution[7]. Let's takes a real world example where problem is solved using MapReduce programming model.

Problem: In election we need to find out which election party got how many votes in every state.

Solution using MapReduce Parallel Programming:

State wise votes will depend on city wise vote count and all cities under one state will together give the total votes from that country.Note that we can calculate total count from one state without caring about votes coming from different cities of other states, we can use parallel algorithm(MapReduce).Here if you start doing sequentially instead of doing parallel, you need to start with empty list of states and then iterate through the vast list of cities and for each city, look at the state, and then update(add) state vote count.You can think from performance point of view also, how bad it would be. Luckily we can use parallel programming here and distribute data set per state wise and then work on each states parallel to calculate total votes coming in from different cities under one state

C. MAPREDUCE FOR DISTRIBUTED SYSTEMS

The distributed computation for MapReduce works in stages, The first stage comprises of splitting input files into various chunks, i.e into 'M' different pieces. Stage 2 will be to fork the processes from master to several worker threads. Stage 3 will be mapping tasks and will comprise of 3 steps:•Reads contents of the input shard assigned to it • Parses key/value pairs out of the input data • Passes each pair to a user-defined map function – Produces intermediate key/value pairs – These are buffered in memory. Stage 4 will be to create intermediate files by partitioning the section into parts using partitioning function. The predecessor stage to partitioning will be sorting followed by reduce stage. In the end the result is returned to the user and the same output is available in 'R' distinct files.

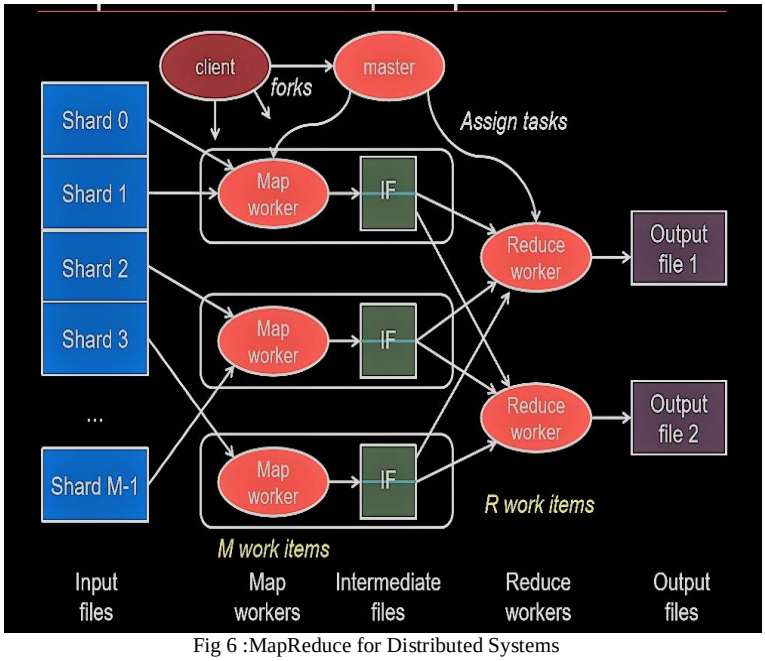

Fig 6 :MapReduce for Distributed Systems

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

V. BIG DATA ANALYTICS

A. *BIG DATA*

Big data is a data which has the potential to be mined using big data handling frameworks. It is data whose scale, distribution,, Diversity, and / or timeliness require the use of new technical architectures and analytics to enable insights that unlock new sources of business values. This data which usually resides in OLAP is characterized by the 3 V's.

Volume – Refers to the size of the data

Variety – Relates to the different types of data i.e structured, semistructured, and unstructured.

Velocity – The speed at which data is being generated

This data is generated from various sources such as social networking, health-care, banking and finance etc. The size of the data is so humongous that the traditional RDBMS fails to process such data. The internet sources from source forge and other sites give us the detailed information about the big data hadoop, which is a data handling framework and also exhibits properties such as fault-tolerance, transparency, openness, etc [8],[9].

B. *ANALYTICS*

Multiple organizations and companies, associations, are realizing gradually that the data which is generated by them can be instead used to improve their business performance. This is possible cause of analytics performed on data. On analytics a user tends to discover deep insights and intrinsic patterns which are beneficial to them in many ways.

The lifecycle of data analytics :

1) Phase 1 : Discovery –It requires the identification of various data sources who's data should be considered for analytics
2) Phase 2 : Data Preparation- The selected data is cleaned and uncovered from errors and noises in this stage.
3) Phase 3 : Model Planning- Identification of various candidate models for clustering, classification, and establishing relationship in the data according to the given goal.
4) Phase 4: Model Building – The model which is cleaned and designed is constructed using various algorithms and methods.
5) Phase 5: Communicate results –After executing the model the analyst is required to compare the outcome of modelling to the criteria established for success and failures.
6) Phase 6: Operationalize – The results obtained from the previous phases are learnt and brought to implement in an organized way.

C. *Big Data Analytics Architecture*

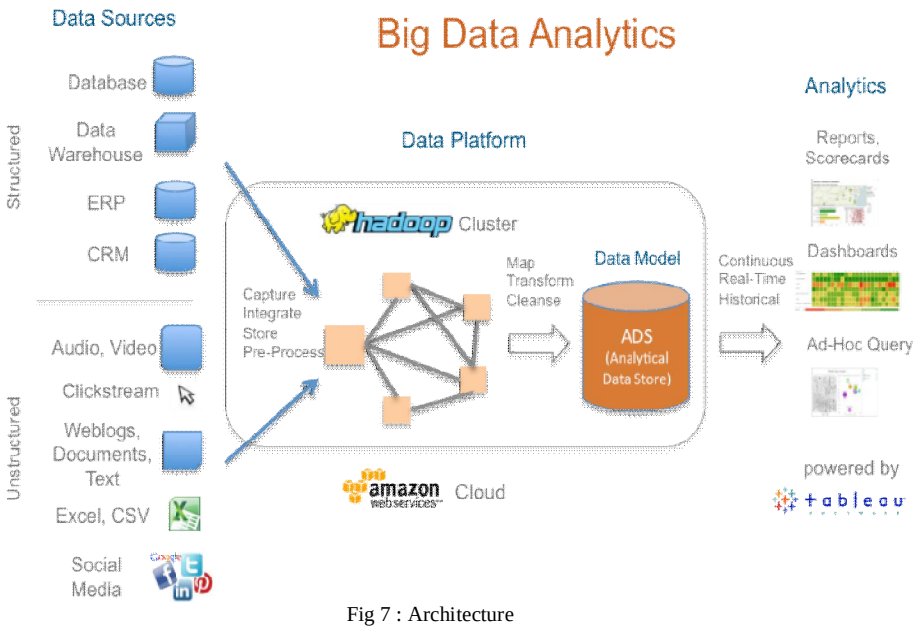

Fig 7 : Architecture

Different categories of data is discovered from various sources. For structured data sources such as RDBMS, Data Warehouse, ERP, CRM of various organizations are considered for analytics whereas for unstructured data audio,

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

video, clickstream, social media, etc is considered. This data is then integrated together followed by storing in a suitable repository and ultimately preprocessed to improve the overall quality of the data. The above steps form the data model which is also called as ADS(Analytical Data Store), finally various analytical tools and algorithms are used to process this data which gives analytics as the outputs. This output is then visualized using various data visualization tools and techniques i.e. Graphs, pie-charts, mosaic plots, treemaps, stick figures, chernofffaces, etc.Z. Xiao and Y. Xiao in their issue of 2014 explained the use of MapReduce for cloud computing[10].

VII. PROPOSED WORK

We propose an e□cient use of MapReduce operations and scheduling on distributed systems. The task components of MapReduce such as Mapper, Practitioners, Combiners,Reducers are described in this paper. The main issues in distributed systems using MapReduce for fast processing is job scheduling. The various issues and solutions are discussed in the later half of the survey. The paper also outlines the architectures of MapReduce implemented in parallel, distributed, and traditional systems.

A. ISSUES AND CHALLENGES

The main issues and challenges of MapReduce related to data storage are the schema-free, index-free terms, and lack of standardized SQL-like language. Grolinger, Katarina et.al in their first edition of 2014 explained the issues related to analytics are scaling complex linear algebra, interactive analysis, iteractive algorithms, and statistical challenges for learning[11]. Other issues faced by MapReduce in distributed systems are online processing, privacy and security. The solutions or the approaches to these discussed issues is efficient communication between the nodes, data pre-processing, security analytics and privacy enforcement.

B. FUTURE SCOPE

In this paper the problem of job scheduling and types of issues in job scheduling is discussed in section 3. The popular issue among all the issue is 'Throughput gain', the throughput gain for the systems can be simply increased by 70% by using efficient system components. The other future works are implementing memory aware scheduling, resource scheduling, intermediate data aware scheduling, using past history for learning job properties, and evaluation using richer benchmarks. MapReduce can be implemented across the distributed systems efficiently by improving the scheduling issues and other security concerns.

VI. CONCLUSION

The MapReduce programming model has been successfully used at Google for many different purposes. First of all the reasons, the model is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault-tolerance, openness, extensibility, scalability, locality optimization, and load balancing. Second, a large variety of problems are easily expressible as MapReduce computations and can be used for sorting, for data mining, for machine learning, and many other systems. Third, we have developed an implementation of MapReduce that scales to large clusters of machines comprising thousands of machines. The implementation makes efficient use of these machine resources and therefore is suitable for use on many of the large computational problems encountered at distributed systems. We have learned several things from this work. First, restricting the programming model makes it easy to parallelize and distribute computations and to make such computations fault-tolerant. Second, network bandwidth is a scarce resource. A number of optimizations in our system are therefore targeted at reducing the amount of data sent across the network. Third, redundant execution can be used to reduce the impact of slow machines, and to handle machine failures and data loss.

REFERENCES

1] Jefferey Dean, Sanjay Ghemawat, "MapReduce: Simplified data processing on large clusters", Google,Inc 2004.
2] Howard Karlo□,SiddharthSuri, Sergei Vassilvitskii, "A Model of Computation for MapReduce",.
3] Hadoop wiki - powered by. http://wiki.apache.org/ hadoop/PoweredBy.
4] Yahoo! partners with four top universities to advance cloud computing systems and applications research. Yahoo! Press Release, 2009. http://research.yahoo. com/news/2743

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 4, Issue 12, December 2016

5] J. Feldman, S. Muthukrishnan, A. Sidiropoulos, C. Stein, and Z. Svitkina. On distributing symmetric streaming computations. In S.-H. Teng, editor, SODA, pages 710–719. SIAM, 2008
6] D. K. G. Campbell. A survey of models of parallel computation. Technical report, University of York, March 1997.
7] L. G. Valiant. A bridging model for parallel computation. CACM, 33(8):103–111, August 1990
8] Apache Hadoop, http://hadoop.apache.org
9] Storm, distributed and fault-tolerant realtime computation, http://storm-project.net/
10] Z. Xiao and Y. Xiao, "Achieving accountable MapReduce in cloud computing," Future Generation Computer Systems, 30,pp. 1-13, 2014.
11] Grolinger, Katarina; Hayes, Michael; Higashino, Wilson A.; L'Heureux, Alexandra; Allison, David S.; and Capretz, Miriam A.M., "Challenges for MapReduce in Big Data" (2014).Electrical and Computer Engineering Publications.Paper 44. http://ir.lib.uwo.ca/electricalpub/44

BIOGRAPHY

Ajinkya Kunjir and Basil Shaikh are computer science engineering students from University of Pune, currently pursuing their final year of engineering from Modern Education Society's College of Engineering, Pune. Our research interests are Big Data, Data Mining, Artificial Intelligence, Machine Learning.