International Journal of scientific research and management (IJSRM)
||Volume||3||Issue||2||Pages|| 2106-2112||2015||
Website: www.ijsrm.in ISSN (e): 2321-3418

Big Data Analysis: Challenges and Solutions

Dipak M. Durgude[1], Nilesh S.Yalij[2]
Department of Computer Engineering
Indira College of Engineering and Management
Parandwadi,Pune,410506
ddurgude19@gmail.com[1] , nileshyalij@gmail.com[2]
Ashwini B. Bhosale[3] , Manisha Bharati[4]
Department of Computer Engineering
Indira College of Engineering and Management
Parandwadi,Pune,410506
ashwinibhosale298@gmail.com
manishabharati@indiraicem.ac.in[4]

Abstract—
We live in on-demand, on-command Digital universe with data prolife ring by Institutions, Individuals and Machines at a very high rate. This data is categories as "Big Data" due to its sheer Volume, Variety, Velocity and Veracity. Most of this data is unstructured, quasi structured or semi structured and it is heterogeneous in nature.

The volume and the heterogeneity of data with the speed it is generated, makes it difficult for the present computing infrastructure to manage Big Data. Traditional data management, warehousing and analysis systems fall short of tools to analyze this data.

Due to its specific nature of Big Data, it is stored in distributed file system architectures. Hadoop and HDFS by Apache is widely used for storing and managing Big Data. Analyzing Big Data is a challenging task as it involves large distributed file systems which should be fault tolerant, flexible and scalable. Map Reduce is widely been used for the efficient analysis of Big Data. Traditional DBMS techniques like Joins and Indexing and other techniques like graph search is used for classification and clustering of Big Data. These techniques are being adopted to be used in Map Reduce. In this research paper the authors suggest various methods for catering to the problems in hand through Map Reduce framework over Hadoop Distributed File System (HDFS). Map Reduce is a Minimization technique which makes use of file indexing with mapping, sorting, shuffling and finally reducing. Map Reduce techniques have been studied at in this paper which is implemented for Big Data analysis using HDFS.

***Keyword*-Big Data Analysis, Big Data Management, Map Reduce, HDFS**

I. Introduction

Big Data encompasses everything from click stream data from the web to genomic and proteomic data from biological research and medicines. 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. Businesses are primarily concerned with managing unstructured data, because over 80 percent of enterprise data is unstructured [26] and require significant storage space and effort to manage. “Big data” refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyses [3].

Big data analytics is the area where advanced analytic techniques operate on big data sets. It is really about two things, Big data and Analytics and how the two have teamed up to create one of the most profound trends in business intelligence (BI) [4]. Map Reduce by itself is capable for analyzing large distributed data sets; but due to the heterogeneity, velocity and volume of Big Data, it is a challenge for traditional data analysis and management tools [1] [2]. A problem with Big Data is that they use NoSQL and has no Data Description Language (DDL) and it supports transaction processing. Also, web-scale data is not universal and it is heterogeneous. For analysis of Big Data, database integration and cleaning is much harder than the traditional mining approaches [4]. Parallel processing and distributed computing is becoming a standard procedure which are nearly non-existent in RDBMS. Map Reduce has following characteristics [12]; it supports Parallel and distributed processing, it is simple and its architecture is shared-nothing which has commodity diverse hardware (big cluster).Its functions are programmed in a high-level programming language (e.g. Java, Python) and it is flexible. Query processing is done through NoSQL integrated in HDFS as Hive tool [20]. Analytics helps to discover what has changed and the possible solutions. Second, advanced analytics is the best way to discover more business opportunities, new customer segments, identify the best suppliers, associate products of affinity, understand sales seasonality[25] etc. Traditional experience in data warehousing, reporting, and online analytic processing (OLAP) is different for advanced forms of analytics [6]. Organizations are implementing specific forms of analytics, particularly called advanced analytics. These are an collection of related techniques and tool types, usually including predictive analytics, data mining, statistical analysis, complex SQL, data visualization, artificial intelligence, natural language processing. Database analytics platforms such as Map Reduce, in-database analytics, in-memory databases, and columnar data stores [6] [9] are used for standardizing them.

With big data analytics, the user is trying to discover new business facts that no one in the enterprise knew before, a better term would be "discovery analytics. To do that, the analyst needs large volumes of data with plenty of detail. This is often data that the enterprise has not yet tapped for analytics example, the log data. The analyst might mix that data with historic data from a data warehouse and would discover for example, new change behavior in a subset of the customer base. The discovery would lead to a metric, report, analytic model, or some other product of BI, through which the company could track and predict the new form of customer behavioral change.

Discovery analytics against big data can be enabled by different types of analytic tools, including those based on SQL queries, data mining, statistical analysis, fact clustering, data visualization, natural language processing, text analytics, artificial intelligence etc. [4-6]. A unique challenge for researchers system and academicians is that the large datasets needs special processing systems [5]. Map Reduce over HDFS gives Data Scientists [1-2] the techniques through which analysis of Big Data can be done. HDFS is a distributed file system architecture which encompasses the original Google File System [13].Map Reduce jobs use efficient data processing techniques which can be applied in each of the phases of Map Reduce; namely Mapping, Combining, Shuffling, Indexing, Grouping and Reducing [7]. All these techniques have been studied in this paper for implementation in Map Reduce tasks

II. Big Data : Opportunities and challenges

In the distributed systems world, "Big Data" started to become a major issue in the late 1990‟s due to the impact of the world-wide Web and a resulting need to index and query its rapidly mushrooming content. Database technology (including parallel databases) was considered for the task, but was found to be neither well-suited nor cost-effective [5] for those purposes. The turn of the millennium then brought further challenges as companies began to use information such as the topology of the Web and users‟ search histories in order to provide increasingly useful search results, as well as more effectively-targeted advertising to display alongside and fund those results. Google's technical response to the challenges of Web-scale data management and analysis was simple, by database standards, but kicked off what has become the modern "Big Data" revolution in the systems world [3]. To handle the challenge of Web-scale storage, the Google File System (GFS) was created [13]. GFS provides clients with the familiar OS-level byte-stream abstraction, but it does so for extremely large files whose content can span hundreds of machines in shared-nothing clusters created using inexpensive commodity hardware [5]. To handle the challenge of processing the data in such large files, Google pioneered its Map Reduce programming model and platform [1][13]. This model, characterized by some as "parallel programming for dummies", enabled Google's developers to process large collections of data by writing two user-defined functions, map and reduce, that the Map Reduce framework applies to the instances (map) and sorted groups of instances that share a common key (reduce) – similar to the sort of partitioned parallelism utilized in shared-nothing parallel query processing.

Driven by very similar requirements, software developers at Yahoo!, Facebook, and other large Web companies followed suit. Taking Google's GFS and Map Reduce papers as rough technical specifications, open-source equivalents were developed, and the Apache Hadoop Map Reduce platform and its underlying file system (HDFS, the Hadoop Distributed File System) were born [1] [12]. The Hadoop system has quickly gained traction, and it is now widely used for use cases including Web indexing, clickstream and log analysis, and certain large-scale information extraction and machine learning tasks. Soon tired of the low-level nature of the Map Reduce programming model, the Hadoop community developed a set of higher-level declarative languages for writing queries and data analysis pipelines that are compiled into Map Reduce jobs and then executed on the Hadoop Map Reduce platform. Popular languages include Pig from Yahoo! [18], Jaql from IBM [28], and Hive from Facebook [18]. Pig is relational-algebra-like in nature, and is reportedly used for over 60% of Yahoos Map Reduce use cases; Hive is SQL-inspired and reported to be used for over 90% of the Facebook Map Reduce use cases. Microsoft's technologies include a parallel runtime system called Dryad and two higher-level programming models, Dryad LINQ and the SQL like SCOPE language [27], which utilizes Dryad under the covers. Interestingly, Microsoft has also recently announced that its future "Big Data" strategy includes support for Hadoop[24].

III. 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 Map Reduce for distributed data processing and is works with structured and unstructured data. Figure1Illustrates the layers found in the software architecture of a Hadoop stack [17] [19]. 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 Map Reduce programming model. High-level language compilers (Pig and Hive) are thus the topmost layer in the Hadoop software stack for such clients.

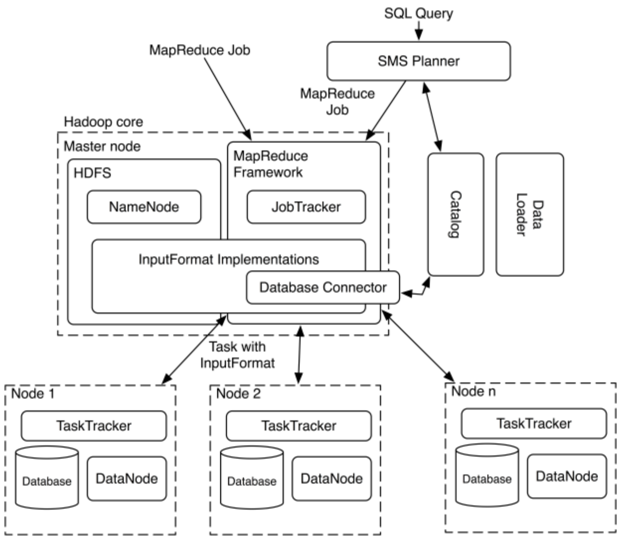

FIGURE 1.HADOOP ARCHITECTURE LAYERS

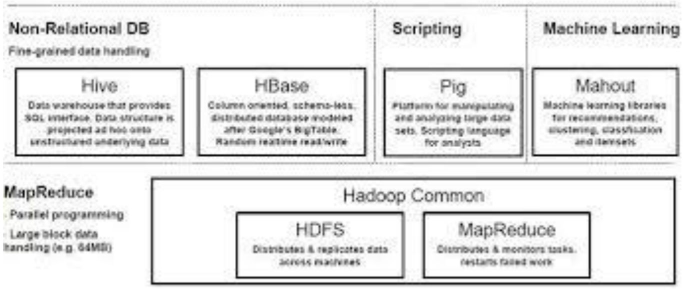

FIGURE 2.HADOOP ARCHITECTURE TOOLS AND USAGE

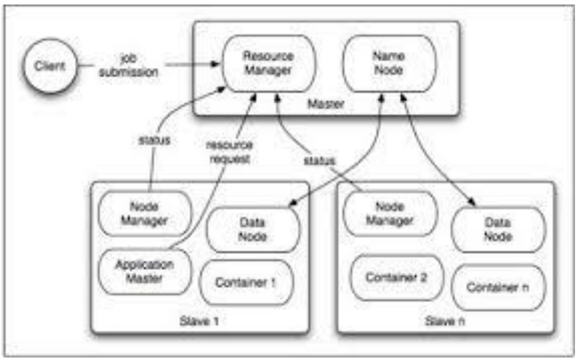

FIGURE 3. HDFS CLUSTERS

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 Map Reduce and Hadoop over HDFS [4] [9]. Figure 3 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. BIG DATA ANALYSIS

Heterogeneity, scale, timeliness, complexity, and privacy problems with Big Data hamper the progress at all phases of the process that can create value from data. Much data today is not natively in structured format; for example, tweets and blogs are weakly structured pieces of text, while images and video are structured for storage and display, but not for semantic content and search: transforming such content into a structured format for later analysis is a major challenge [15]. The value of data enhances when it can be linked with other data, thus data

integration is a major creator of value. Since most data is directly generated in digital format today, we have the opportunity and the challenge both to influence the creation to facilitate later linkage and to automatically link previously created data. Data analysis, organization, retrieval, and modeling are other foundational challenges [6

Figure 4, below gives a glimpse of the Big Data analysis tools which are used for efficient and precise data analysis and management jobs. The Big Data Analysis and management setup can be understood through the layered structured defined in the figure. The data storage part is dominated by the HDFS distributed file system architecture; other mentioned architectures available are Amazon Web Service (AWS) [23], Hbase and Cloud Store etc. The data processing tasks for all the tools is Map Reduce; we can comfortably say that it is the de-facto Data processing tool used in the Big Data paradigm.

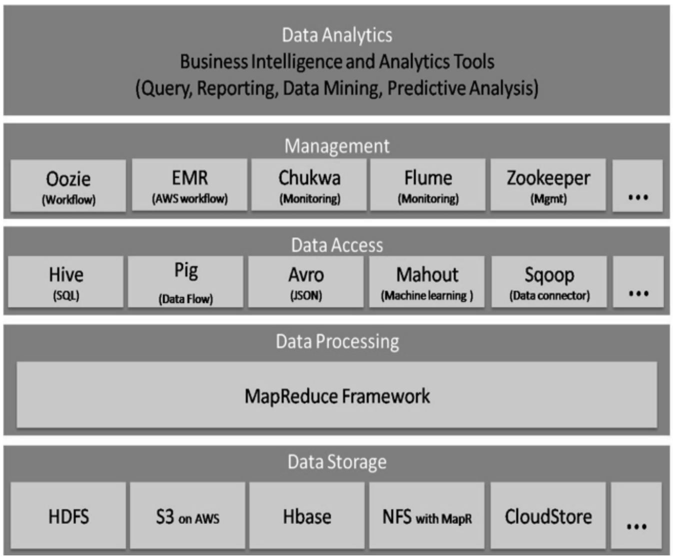

Figure 4. Big Data Analysis Tools

For handling the velocity and heterogeneity of data, tools like Hive, Pig and Mahout are used which are parts of Hadoop and HDFS framework. It is interesting to note that for all the tools used, Hadoop over HDFS is the underlying architecture. Oozie and EMR with Flume and Zookeeper are used for handling the volume and veracity of data, which are standard Big Data management tools. The layer with their specified tools forms the bedrock for Big Data management and analysis framework.

V. MAP REDUCE

Map Reduce [1-2] is a programming model for processing large-scale datasets in computer clusters. The Map Reduce 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 Map Reduce 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) The signatures of map() and reduce() are as follows : **map (k1,v1) ! list(k2,v2)and reduce (k2,list(v2)) ! list(v2)**

A Map Reduce cluster employs a master-slave architecture where one master node manages a number of slave nodes [19]. In the Hadoop, the master node is called Job Tracker and the slave node is called Task Tracker as shown in the figure 7. Hadoop launches a Map Reduce job by first splitting the input dataset into even-sized data blocks. Each data block is then scheduled to one Task Tracker node and is processed by a map task. The Task Tracker node notifies the Job Tracker when it is idle. The scheduler then assigns new tasks to it. The scheduler takes data locality into account when it disseminates data blocks.

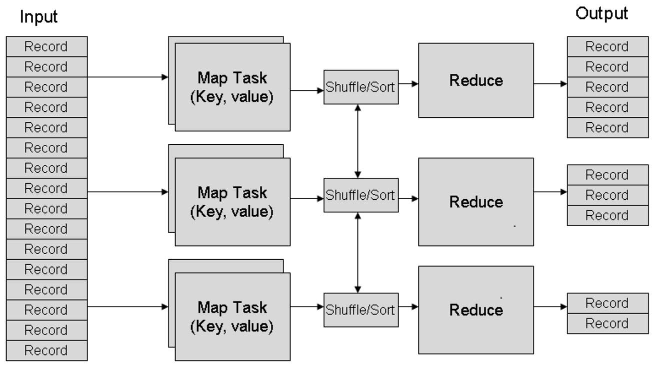

Figure 5. Map Reduce Architecture and Working

It always tries to assign a local data block to a Task Tracker. If the attempt fails, the scheduler will assign a rack-local or random data block to the Task Tracker 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 i has inbuilt processes for status and monitoring of heterogeneous and large datasets as in Big Data [14].

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

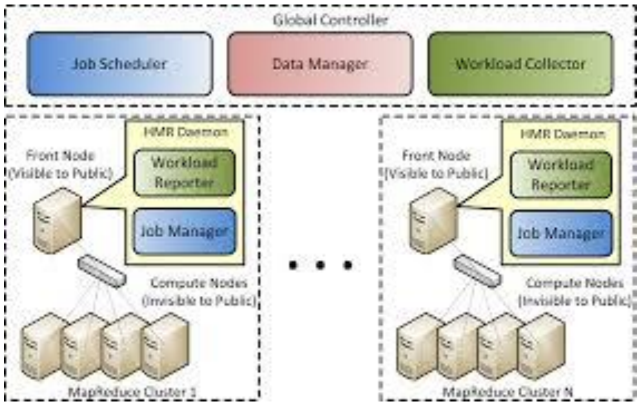

Figure 6.Map Reduce Components

B. Map Reduce Working

We implement the Mapper and Reducer interfaces to provide the map and reduce methods as shown in figure 6. 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 [19]. 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 block size of 128MB, you'll end up with 82,000 maps [17] [19].

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.

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[19].

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) Partitioned

Partitioned partitions the key space. Partitioned 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 Partitioned is the default Partitioned.

b) Reporter

Reporter is a facility for Map Reduce 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 Map Reduce 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 practitioners.

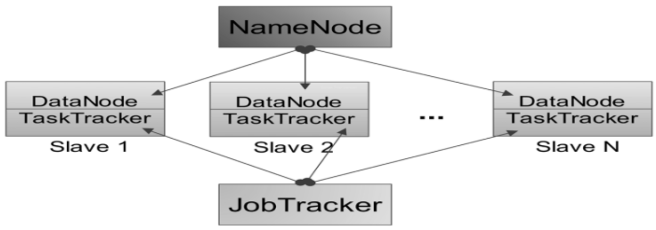

Figure 7. Map Reduce Working through Master / Slave

C. Map Reduce techniques

- Combining

Combiners provide a general mechanism within the Map Reduce 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

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[19].

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 Map Reduce 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 Map Reduce framework to collect data output by the Mapper or the Reducer (either the intermediate outputs or the

A. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semi-/colons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is given by Young [7].

VI. Conclusion

The need to process enormous quantities of data has never been greater. Not only are terabyte- and petabyte-scale datasets rapidly becoming commonplace, but there is consensus that great value lies buried in them, waiting to be unlocked by the right computational tools. In the commercial sphere, business intelligence, driven by the ability to gather data from a dizzying array of sources. 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 [6]. For engineers building information processing tools and applications, large and heterogeneous datasets which are generating continuous flow of data, lead to more effective algorithms for a wide range of tasks, from machine translation

to spam detection. In the natural and physical sciences, the ability to analyses massive amounts of data may provide the key to unlocking the secrets of the cosmos or the mysteries of life. MapReduce can be exploited to solve a variety of problems related to text processing at scales that would have been unthinkable a few years ago [15]. No tool no matter how powerful or flexible can be perfectly adapted to every task. There are many examples of algorithms that depend crucially on the existence of shared global state during processing, making them difficult to implement in MapReduce (since the single opportunity for global synchronization in MapReduce is the barrier between the map and reduce phases of processing). Implementing online learning algorithms in MapReduce is problematic [14]. The model parameters in a learning algorithm can be viewed as shared global state, which must be updated as the model is evaluated against training data. All processes performing the evaluation (presumably the mappers) must have access to this state. In a batch learner, where updates occur in one or more reducers (or, alternatively, in the driver code), synchronization of this resource is enforced by the MapReduce framework. However, with online learning, these updates must occur after processing smaller numbers of instances. This means that the framework must be altered to support faster processing of smaller datasets, which goes against the design choices of most existing MapReduce implementations. Since MapReduce was specifically optimized for batch operations over large amounts of data, such a style of computation would likely result in insufficient use of resources [2]. In Hadoop, for example, map and reduce tasks have considerable start-up costs.

VII. Advancements

Streaming algorithms [9] represent an alternative programming model for dealing with large volumes of data with limited computational and storage resources. This model assumes that data are presented to the algorithm as one or more streams of inputs that are processed in order, and only once. Stream processing is very attractive for working with time-series data (news feeds, tweets, sensor readings, etc.), which is difficult in MapReduce (once again, given its batch-oriented design). Another system worth mentioning is Pregel [16], which implements a programming model inspired by Valiant's Bulk Synchronous Parallel (BSP) model. Pregel was specially designed for large-scale graph algorithms, but unfortunately there are few published details at present.

Pig [28], which is inspired by Google [13], can be described as a data analytics platform that provides a lightweight scripting language for manipulating large datasets. Although Pig scripts (in a language called Pig Latin) are ultimately converted into Hadoop jobs by Pig's execution engine through joins, allow developers to specify data transformations (filtering, joining, grouping, etc.) at a much higher level. Similarly, Hive [20], another open-source project, provides an abstraction on top of Hadoop that allows users to issue SQL queries against large relational datasets stored in HDFS. Hive queries, in HiveQL are compiled down to Hadoop jobs by the Hive query engine. Therefore, the system provides a data analysis tool for users who are already comfortable with relational databases, while simultaneously taking advantage of Hadoop's data processing capabilities [11]. The power of MapReduce derives from providing an abstraction that allows developers to harness the power of large clusters but abstractions manage complexity by hiding details and presenting well-defined behaviors to users of those abstractions. This process makes certain tasks easier, but others more difficult, if not impossible. MapReduce is certainly no exception to this generalization, even within the Hadoop/HDFS/MapReduceecosystem; it is already observed the development of alternative approaches for expressing distributed computations. For example, there can be a third merge phase after map and reduce to better support relational operations. Join processing mentioned n the paper can also tackle the Map Reduce tasks effectively. The future directions in Big Data analysis gives a very encouraging picture as the tools are build on the existing paradigm of HDFS and Hadoop, overcoming the existing drawback of the present systems and the advantages it provides over the traditional data analysis tools.

REFERENCES

[1] Jefry Dean and Sanjay Ghemwat, MapReduce:A Flexible Data Processing Tool, Communications of the ACM, Volume 53, Issuse.1,January 2010, pp. 72-77.

[2] Jefry Dean and Sanjay Ghemwat,.MapReduce: Simplified data processing on large clusters, Communications of the ACM, Volume 51 pp. 107–113, 2008

[3] Brad Brown, Michael Chui, and James Manyika, Are you ready for the era of „big data‟?,McKinseyQuaterly,Mckinsey Global Institute, October 2011.

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

[5] MarcinJedyk, MAKING BIG DATA, SMALL, Using distributed systems for processing, analysing and managing large huge data sets, Software Professional‟s Network, Cheshire Data systems Ltd.

[6] OnurSavas, YalinSagduyu, Julia Deng, and Jason Li,Tactical Big Data Analytics: Challenges, Use Cases and Solutions, Big Data Analytics Workshop in conjunction with ACM Sigmetrics 2013,June 21, 2013.

[7] Kyuseok Shim, MapReduce Algorithms for Big Data Analysis, DNIS 2013, LNCS 7813, pp. 44–48, 2013.

[8] Raja.Appuswamy,ChristosGkantsidis,DushyanthNarayanan,OrionHodson,AntonyRowstron, Nobody ever got fired for buying a cluster, Microsoft Research, Cambridge, UK, Technical Report,MSR-TR-2013-2

[9] Carlos Ordonez, Algorithms and Optimizations for Big Data Analytics: Cubes, Tech Talks,University of Houston, USA.

[10] Spyros Blanas, Jignesh M. Patel,VukErcegovac, Jun Rao,Eugene J. Shekita, YuanyuanTian, A Comparison of Join Algorithms for Log Processing in MapReduce, SIGMOD‟10, June 6–11, 2010, Indianapolis, Indiana, USA.

[11] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein,JohnGerth, Justin Talbot,KhaledElmeleegy, Russell Sears, Online Aggregation and Continuous Query support in MapReduce, SIGMOD‟10, June 6–11, 2010, Indianapolis, Indiana, USA.

[12] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in USENIXSymposium on Operating Systems Design and Implementation, San Francisco, CA, Dec. 2004, pp. 137–150.

[13] S. Ghemawat, H. Gobioff, and S. Leung, "The Google File System." in ACM Symposium on Operating Systems Principles, Lake George, NY, Oct 2003, pp. 29 – 43.

[14] HADOOP-3759: Provide ability to run memory intensive jobs without affecting other running tasks on the nodes. https://issues.apache.org/jira/browse/HADOOP-3759

[15] VinayakBorkar, Michael J. Carey, Chen Li, Inside "Big Data Management":Ogres, Onions, or Parfaits?, EDBT/ICDT 2012 Joint Conference Berlin, Germany,2012 ACM 2012, pp. 3-14.

[16] GrzegorzMalewicz, Matthew H. Austern, Aart J. C. Bik, James C.Dehnert, Ilan Horn, NatyLeiser, and GrzegorzCzajkowski,Pregel: A System for Large-Scale Graph Processing, SIGMOD‟10, June 6–11, 2010, pp. 135-145.

[17] Hadoop,"PoweredbyHadoop,"http://wiki.apache.org/hadoop/PoweredBy.

[18] PIGTutorial,YahooInc., http://developer.yahoo.com/hadoop/tutorial/pigtutorial.html

[19] Apache: Apache Hadoop, http://hadoop.apache.org

[20] Apache Hive, http://hive.apache.org/

[21] Apache Giraph Project, http://giraph.apache.org/

[22] Mahout, http://lucene.apache.org/mahout/

[23] Amazon Simple Storage Service (Amazon S3). http://aws.amazon.com/s3/

[24] Windows.Azure.Storage.http://www.microsoft.com/windowsazure/features/storage/

[25] The Age of Big Data. Steve Lohr. New York Times, Feb 11, 2012. http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html

[26] Information System & Management, ISM Book, 1st Edition 2010, EMC2, Wiley Publishing

[27] Dryad - Microsoft Research, http://research.microsoft.com/en-us/projects/dryad/

[28] IBM-What.is.Jaql, www.ibm.com/software/data/infosphere/hadoop/jaql/