Hadoop Mapreduce Framework in Big Data Analytics

Vidyullatha Pellakuri[1], Dr.D. Rajeswara Rao[2]
[1](Research Scholar, Department of CSE, KL University, Guntur, India)
[2](Professor, Department of CSE, KL University, Guntur, India)

ABSTRACT: *As Hadoop is a Substantial scale, open source programming system committed to adaptable, disseminated, information concentrated processing. Hadoop [1] Mapreduce is a programming structure for effectively composing requisitions which prepare boundless measures of information (multi-terabyte information sets) in-parallel on extensive bunches (many hubs) of merchandise fittings in a dependable, shortcoming tolerant way. A Mapreduce [6] skeleton comprises of two parts. They are "mapper" and "reducer" which have been examined in this paper. Fundamentally this paper keeps tabs on Mapreduce modifying model, planning undertakings, overseeing and re-execution of the fizzled assignments. Workflow of Mapreduce is indicated in this exchange.*

Keywords *– Framework, HDFS, Mapreduce, Shuffle, Workflow.*

I. INTRODUCTION

Hadoop was designed especially for the analysis of large data sets to build scalable, distributed applications. To manage sizably voluminous data, Hadoop[2] implements the paradigm called MapReduce defined by Google according to which the applications are divided into minute chunks of software, each of which can be run on a distinct node of all those who make up the system. Companies like Amazon, Cloudera, IBM, Intel, Twitter, Facebook and others are formulate their immensely enormous data message and providing insight into where the market is headed utilizing Apache Hadoop technology. MapReduce is a programming model designed for processing immensely colossal volumes of data in parallel by dividing the work into a set of independent tasks. MapReduce [3] programs are indited in a particular style influenced by functional programming constructs, categorically idioms for processing lists of data. A MapReduce program is possessed of a "Map()" procedure that executes filtering and sorting (for example sorting people by first name into queues, one queue for each one name)and a "Reduce()" procedure that implements a synopsis operation (such as counting the number of students in each queue, yielding name by uniformity). The "MapReduce Framework" (likewise called "infrastructure" or "framework") organizes by assembling the distributed servers, running the various tasks in parallel, controlling all communications and data transfers between the numerous parts of the framework, and accommodating for redundancy and fault tolerance. MapReduce [4] libraries have been accounted in numerous programming languages, with distinct levels of optimization. A prevalent open-source requisition is Apache Hadoop.

II. HADOOP MAPREDUCE

MapReduce is a programming model and software framework first developed by Google (Google's MapReduce paper submitted in 2004) Proposed to facilitate and reduce the processing of vast amounts of data in parallel on enormous clusters of commodity hardware in a reliable, fault-tolerant manner. A MapReduce [5] job typically splits the input data-set into separate blocks which are took care by the map tasks in a absolutely parallel ways. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are saved in a file-system. The framework tends of scheduling tasks, monitoring them and re-executes the failed tasks.

2.1 MAPREDUCE FRAMEWORK

The mapreduce framework consists of two steps namely Map step and reduce step. Master node takes large problem input and slices it into smaller sub problems and distributes these to worker nodes. Worker node may do this again and leads to a multi-level tree structure .Worker processes smaller problem and hands back to master. In Reduce step Master node takes the answers to the sub problems and combines them in a predefined way to get the output/answer to original problem. The MapReduce framework is fault-tolerant because each node in the cluster is expected to report back periodically with completed work and status updates. If a node remains silent for longer than the expected interval, a master node makes note and re-assigns the work to other nodes.

2.2 WORKFLOW IN MAPREDUCE

The key to how MapReduce [6] works is to take input as, conceptually, a list of records. The records are split among the different computers in the cluster by Map. The result of the Map computation is a list of key/value pairs. Reducer then takes each set of values that has the same key and combines them into a single value. So Map takes a set of data chunks and produces key/value

pairs and Reduce merges things, so that instead of a set of key/value pair sets, you get one result. You can't tell whether the job was split into 100 pieces or 2 pieces. MapReduce isn't intended to replace relational databases. It's intended is to provide a lightweight way of programming things so that they can run fast by running in parallel on a lot of machines.

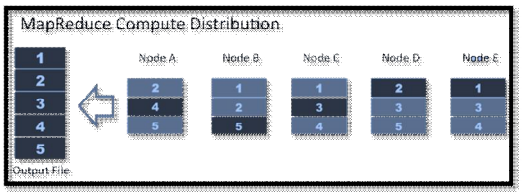

Fig. 1 Computation of MapReduce

MapReduce is important because it allows ordinary developers to use MapReduce library routines to create parallel programs without having to worry about programming for intra-cluster communication, task monitoring or failure handling. It is useful for tasks such as data mining, log file analysis, financial analysis and scientific simulations. Several implementations of MapReduce are available in a variety of programming languages, including Java, C++, Python, Perl, Ruby, and C. Typical Hadoop cluster integrates MapReduce and HFDS with master / slave architecture which consists of a Master node and multiple slave nodes. Master node contains Job tracker node (MapReduce layer), Task tracker node (MapReduce layer), Name node (HFDS layer) ,Data node (HFDS layer). Multiple slave nodes are Task tracker node (MapReduce layer) and Data node (HFDS layer). MapReduce layer has job and task tracker nodes while HFDS layer has name and data nodes [11].

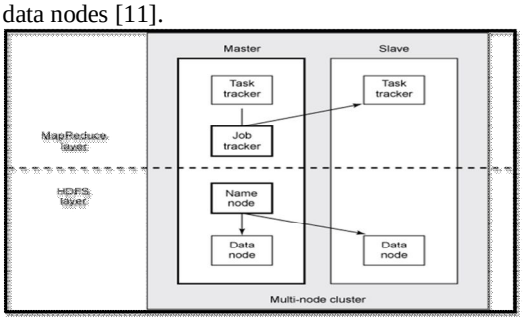

Fig. 2 Layers in MapReduce

Although the Hadoop framework is implemented in Java TM, MapReduce applications need not be written in Java. Hadoop Streaming is a utility which allows users to create and run jobs with any executables (e.g. shell utilities) as the mapper and/or the reducer. Hadoop Pipes is a SWIG-compatible C++ API to implement MapReduce applications (non JNITM based).

2.3 MAPREDUCE FUNCTIONALITY

The mapreduce functionality depends on mapper, shuffler and reducer. The 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. All intermediate values associated with a given output key are subsequently grouped by the framework, and passed to the Reducer(s) to determine the final output. The Mapper [16] outputs are sorted and then partitioned per Reducer. The total number of partitions is the same as the number of reduce tasks for the job. The intermediate, sorted outputs are always stored in a simple (key-length, key, and value-length) format. The 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. 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

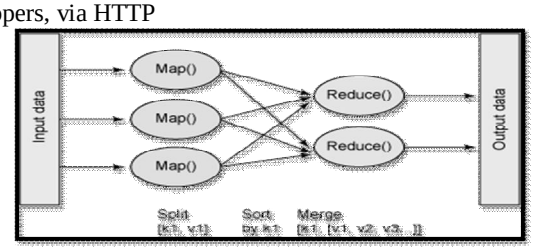

Fig. 3 Fucntionality of MapReduce

The framework [7] 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. If equivalence rules for grouping the intermediate keys are required to be different from those for grouping keys before reduction and it can be used to control how intermediate keys are grouped, these can be used in conjunction to simulate secondary sort on values. In this phase the 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. The output of the Reducer is not sorted.

2.3.1 NUMBER OF MAPS AND REDUCES

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. Thus, if you expect 10TB of input data and have a block size of 128MB, you'll end up with 82,000 maps[15], The right number of reduces seems to be 0.95 or 1.75 multiplied by (no. of nodes* mapred .task tracker. reduce. tasks. maximum). With 0.95 all of the reduces can launch immediately and start transferring map[16] 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. 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.

2.4 DATA ORGANIZATION

In many organizations, Hadoop and other MapReduce solutions are only the examples in the larger data analysis platform. Data will typically have to be translated in order to interface perfectly with the other organizations. Similarly, the data might have to be transmuted from its original state to a new state to clear analysis in MapReduce easier.

2.4.1 NATURE OF DATA

MapReduce systems such as Hadoop aren't being utilized exactly for text analysis anymore. An increasing number of users are deploying MapReduce jobs that analyze data once thought to be excessively difficult for the paradigm. One of the most obvious trends in the nature of data is the boost of image, audio, and video analysis. This kind of data is a serious prospect for a distributed system using MapReduce [10] because these files are typically very big. Retailers want to examine their security video to detect what stores are most engaged. Medical imaging analysis is becoming harder with the astronomical resolutions of the image. Videos have colored pixels that change over time, laid out a flat grid. The data are analyzed in order by challenging to take a look at 10-pixel by 10-pixel by 5-second section of video and audio as a "record." As multidimensional data increases in popularity, there are more patterns showing how to logically separate the data into records and input splits properly. For example, SciDB, an open-source analytical database, is specifically built to deal with multi-dimensional data. MapReduce is traditionally a batch analytics system, but streaming analytics feels like a natural onward motion. In many production MapReduce systems, data are always streaming in and then gets treated in batch on an interval. For instance, data from web server logs are streaming in, but the MapReduce [17]job is only done every hour. This is inconvenient for a few reasons. First, processing an hour's worth of data at once can strain resources. Novel systems that deal with streaming data in Hadoop have cropped up, most notably the commercial product like HStreaming and the open-source Storm platform, recently released by Twitter.

III.INPUTS AND OUTPUTS

The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs[15] and produces a output of the job as set of <key, value> pairs conceivably of distinct types. The key and value classes have to be serializable by the framework and hence need to implement the writable interface. Additionally, the key classes have to implement the Writable Comparable interface to facilitate sorting by the framework[13].

3.1 I/O TYPES OF A MAPREDUCE JOB

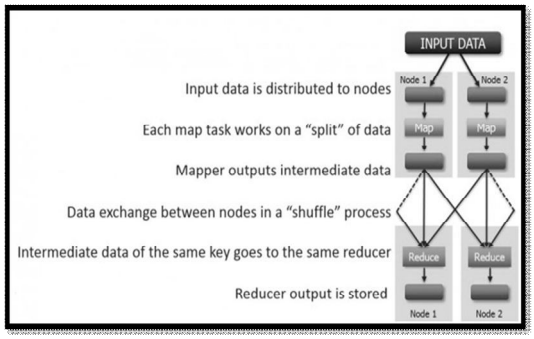

Fig.4 Distribution of Input data

A simple MapReduce program can be written to determine how many times different words appear in a set of files for example if we the files like file1, file2, and file 3.

Input:

File1: Deer, Bear, River
File2: Car, Car, River
File3: Deer, Car, Bear

We can a write a program in mapreduce by using three operations like map, combine, reduce to compute the output.

The first step is Map Step:

First Map	Second Map	Third Map
<Deer,1>	<Car,1>	<Deer,1>
<Bear,1>	<Car,1>	<Car,1>
<River,1>	<River,1>	<Bear,1>

The secondary step is Combine Step:

<Bear,1>	<Car,1>	<Deer,1>	<River,1>
<Beer,1>	<Car,1>	<Deer,1>	<River,1>
	<Car,1>		

The final step is Reduce Step:

<Beer,2> <Car,3> <Deer,2> <River,2>

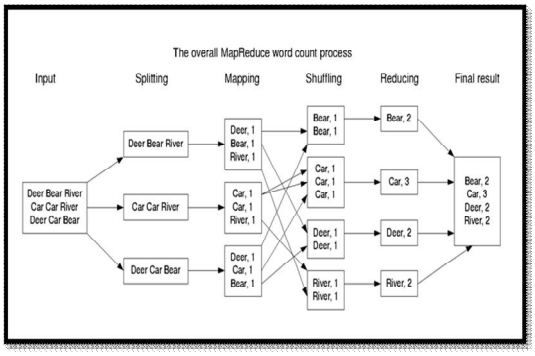

Fig.5 Example showing MapReduce Job

3.2 TASK EXECUTION AND ENVIRONMENT

TaskTracker executes Mapper/Reducer task as a child process in a separate JVM (Java Virtual Machine). The Child task inherits the environment of the parent TaskTracker [12]. A User can specify environmental variables controlling memory, parallel computation settings, segment size. Requirements of applications using MapReduce specifies the Job configuration, input/output locations. It supply map and reduce functions via implementations of appropriate interfaces and/or abstract classes.

3.3 SCHEDULING

Usually, Hadoop uses FIFO to schedule jobs. The scheduler option depends on capacity and fair. Jobs are submitted to the queue according to their priority. Queues are allocated according to the resources capacity. Free resources are allocated to queues away from their total capacity [9][10].

IV.PROMINENT USERS

Lots of companies! such as Yahoo!, AOL, eBay, Facebook, IBM, Last.fm, LinkedIn, The New York Times, Ning, Twitter, and more. In 2007 IBM and Google [14] announced an initiative to use Hadoop to support university courses in distributed computer programming. In 2008 this collaboration and the Academic Cloud Computing Initiative were funded by the NSF and produced the Cluster Exploratory Program (CLuE).

4.1AMAZON'S NEW ELASTIC MAPREDUCE

Amazon has afford a advanced web service which implement Hadoop services as Elastic Map Reduce. Map Reduce is a method that has the capability of breaking a job into hundreds or even thousands of independent parallel processes. A simple process (like counting the words in a book) is broken up into numerous running parts (i.e., The Map), then collect all parts into summary counts (i.e., The Reduce). This grants a programmer to process immensely large data sets in a up-to-date manner.

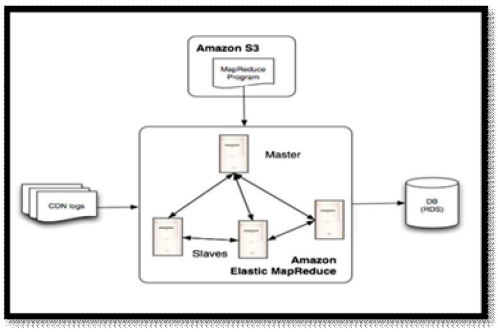

Fig.6 Amazon's Map Reduce Job

The Elastic Map Reduce [19] service is immensely simple to install. Basically, you upload your application to process to an S3(Simple Storage Service) bucket. Then You create a job that locates on S3 of your input and output data sets and your map reduce program. You then also configure the number of EC2 [20] instances you want to run for the Map Reduce job. You can also add some advanced arguments and use more complex processing methods, if you choose. Hadoop runs as a cluster of machines with a specific file system called the HDFS (Hadoop File System). It also has a number of servers called Datanodes (these are the job processing clients) and a master server called the Namenode server. Here is a diagram of an HDFS environment.

V. CONCLUSION

Hadoop MapReduce is a broad scale, open source software framework devoted to scalable, distributed, data-intensive computing. The mapreduce framework breaks up large data into smaller parallelizable chunks and handles scheduling. If you can rewrite algorithms into Maps and Reduces, and your problem can be broken up into small pieces solvable in parallel, then Hadoop's MapReduce is the way to go for a distributed problem solving approach to large datasets. Mapreduce framework is Fault tolerant, decisive and supports thousands of nodes and petabytes of data. The future trend in big data is Apache Hadoop2. It is the second iteration of the Hadoop framework for distributed data processing. Haddop2 adds support for running non-batch applications as well as advanced features are designed to improve system availability.

REFERENCES

[1] White, Tom (10 May 2012). *Hadoop: The Definitive Guide*. O'Reilly Media. p. 3. ISBN 978-1-4493-3877-0.

[2] "Applications and organizations using Hadoop". Wiki.apache.org. 2013-06-19. Retrieved 2013-10-17.

[3] "HDFS User Guide". Hadoop.apache.org. Retrieved 2012-05-23.

[4] "HDFS Architecture". Retrieved 1 September 2013.

[5] "Improving MapReduce performance through data placement in heterogeneous Hadoop Clusters" (PDF). Eng.auburn.ed. April 2010.

[6] "HDFS Users Guide - Rack Awareness". Hadoop.apache.org. Retrieved 2013-10-17.

[7]"Cloud analytics: Do we really need to reinvent the storage stack?". IBM. June 2009.

[8]"HADOOP-6330: Integrating IBM General Parallel File System implementation of Hadoop File system interface". IBM. 2009-10-23.

[9] "Refactor the scheduler out of the JobTracker". Hadoop Common. Apache Software Foundation. Retrieved 9 June 2012.

[10] M. Tim Jones (6 December 2011). "Scheduling in Hadoop". ibm.com. IBM. Retrieved 20 November 2013.

[11]"Under the Hood: Hadoop Distributed File system reliability with Namenode and Avatarnode". Facebook. Retrieved 2012-09-13.

[12] "Under the Hood: Scheduling MapReduce jobs more efficiently with Corona". Facebook. Retrieved 2012-11-9.

[13] "Zettaset Launches Version 4 of Big Data Management Solution, Delivering New Stability for Hadoop Systems and Productivity Boosting Features | | Zettaset.comZettaset.com". Zettaset.com. 2011-12-06. Retrieved 2012-05-23.

[14] Curt Monash. "More patent nonsense — Google MapReduce". dbms2.com. Retrieved 2010-03-07.

[15] D. Wegener, M. Mock, D. Adranale, and S. Wrobel, "Toolkit-Based High-Performance Data Mining of Large Data on MapReduce Clusters," Proc. Int'l Conf. Data Mining Workshops (ICDMW '09), pp. 296-301, 2009

[16] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," in Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6, ser. OSDI'04.

[17] http://www.slideshare.net/mcsrivas/design-scale-and-performance-of-maprs-distribution-for-hadoop

[18] http://www.mapr.com/products/mapr-editions

[19] http://aws.amazon.com/elasticmapreduce/mapr/

[20] http://aws.amazon.com/elasticmapreduce/