ISSN 23499842(Online), Volume 1, Special Issue 1(NCFCSIT 15), March 2015
International Journal of Innovative Trends and Emerging Technologies

HADOOP'S MAP REDUCE FRAME WORKS ADVANTAGES AND CHALLENGES IN DATA ANALYTICS FOR BIG DATA

Author:R.R. KARTHIKEYAN, Research-Scholar,
Bharath University, Chennai-600126
rrkarthikeyan0711@gmail.com
Co-Author & Supervisor: DR.B.RAGHU, Professor & Dean
Department of Computer Science & Engineering
Sri.Ramanujar Engineering College, Chennai – 600127
raghubalraj@gmail.com

Abstract: "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.

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.

In this research paper various methods for catering to the problems in hand through Map Reduce framework over Hadoop Distributed File System (HDFS).

These techniques are being adopted to be used in Map Reduce. 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.

1. Introduction:

Big data analytics is the process of examining large data sets containing a variety of data types to uncover hidden patterns, unknown correlations, market trends,

customer preferences and other useful business information. The analytical findings can lead to more effective marketing, new revenue opportunities, better customer service, improved operational efficiency, competitive advantages over rival organizations and other business benefits.

The primary goal of big data analytics is to help companies make more informed business decisions by enabling data scientists, predictive modelers and other analytics professionals to analyze large volumes of transaction data, as well as other forms of data that may be untapped by conventional business intelligence (BI) programs. That could include Web server logs and Internet click stream data, social media content and social network activity reports, text from customer emails and survey responses, mobile-phone call detail records and machine data captured by sensors connected to the Internet of Things. Some people exclusively associate big data with semi-structured and unstructured data of that sort, but consulting firms like Gartner Inc. and Forrester Research Inc. also consider transactions and other structured data to be valid components of big data analytics applications.

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 etc.

Traditional experience in data warehousing, reporting, and online analytic processing (OLAP) is different for advanced forms of analytics. 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 MapReduce, in-database analytics, in-memory databases, and columnar data stores are used for standardizing them.

2. HADOOP AND HDFS (New File system for Hadoop):

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.

Figure1, Illustrates the layers found in the software architecture of a Hadoop 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.

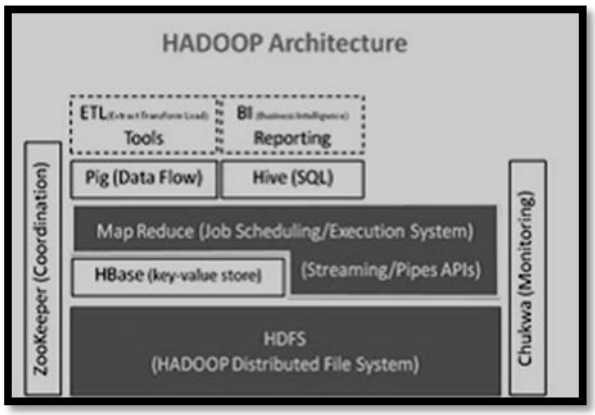
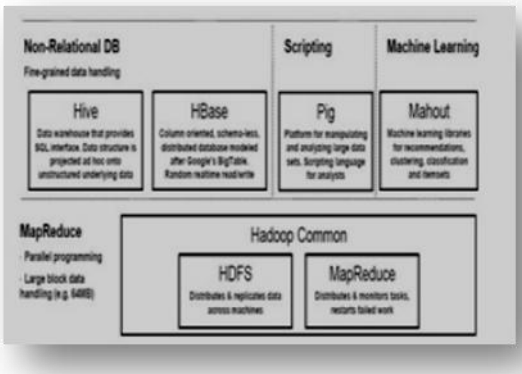


Figure1: Hadoop Architecture Layer

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 figure2, 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.

Figure2: Hadoop Architecture Tools and usage

-

3. MAP REDUCE Framework:

Map Reduce 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 Reduce Components shown in Figure3:

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

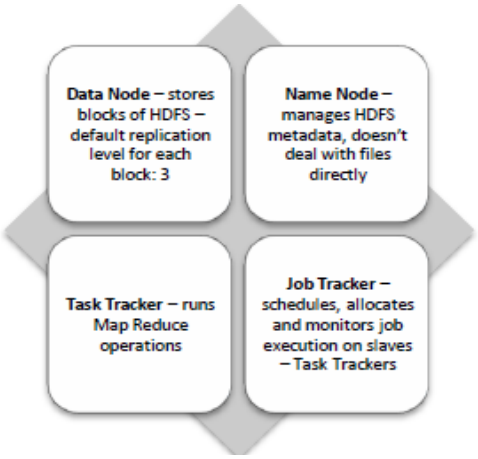


Figure3: Map Reduce Components

.

How Map Reduce works:

3.1) 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 give in 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

Master slave principal shown in Figure4.

.

3.2) 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.

3.2.1) 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 HTT

3.2.2) 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.

3.2.3) 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).

3.2.4) 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.

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 reduces lots 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 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

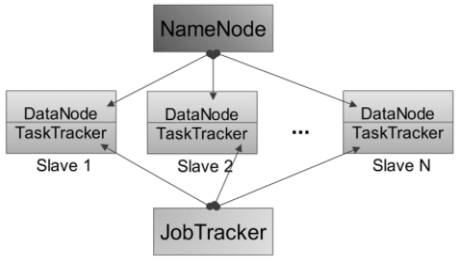


Figure4: Map Reduce Working through Master / Slave

4. 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. 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 analyse 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. 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. 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. In Hadoop, for example, map and reduce tasks have considerable start-up costs.

REFERENCES

[1], Puneet Singh Duggal,Sanchita Paul, International Conference on Cloud, Big Data and Trust 2013, Department of Computer Science & Engineering Birla Institute of Technology, Mesra, Ranchi, India.

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

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

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

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

[7] 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.

[8] 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.

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

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

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