Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 1, Number 9; October, 2014 pp. 93-97

http://www.krishisanskriti.org/jbaer.html

Bundling Hadoop & Map Reduce for Data-Intensive Computing in Distributed Systems

Sanchita Kadambari[1], Kalpana Jaswal[2], Praveen Kumar[3], Seema Rawat[4]

[1,2]M.Tech (CS&E), Amity University, Noida
[3,4]Amity University, Noida

Abstract: **We are living in an age when an explosive amount of data is being generated every day. Data from sensors, mobile devices, social networking websites, scientific data & enterprises – all are contributing to this huge explosion in data. This sudden bombardment can be grasped by the fact that we have created a vast volume of data in the last two years. Big Data- as these large chunks of data is generally called- has become one of the hottest research trends today. Research suggests that tapping the potential of this data can benefit businesses, scientific disciplines and the public sector – contributing to their economic gains as well as development in every sphere. The need is to develop efficient systems that can exploit this potential to the maximum, keeping in mind the current challenges associated with its analysis, structure, scale, timeliness and privacy. The answer lies in Hadoop.**

Keywords: **big data, data revolution, analysis, Hadoop, Map Reduce**

1. INTRODUCTION

Big Data is the hottest trend in the business and IT world right now. We are living in the age of big data where due to the rapid development in the computational power and the WWW, we are producing an overwhelming amount of data, which has led to the need of a change in the existing architectures and mechanisms of the data processing systems. Big data- as these large chunks of data is generally called has redefined the current data processing scenario. The changes in the Web have been defined by analysts such as Gartner and others for describing Big Data as:

- Velocity – how fast the data is entering the systems
- Variety – includes all types of structured and unstructured data
- Volume – the potential data capacity of terabytes to petabytes
- Complexity – includes everything from transferring operational data to big data platforms and the trouble with managing the data across many geographies and locations.

From consumers to companies, people have an unquenchable appetite for data and all that can be done with it. Not only are we relying on data for movie suggestions and gift recommendations but are depending on data for multidisciplinary climate and energy research, building adaptable roads and buildings, better foresighted healthcare, new ways to identify fraud, and keeping a check on consumer behaviour and sentiment. It's a data feast and is not going to end any time soon.

In the past, enterprise systems used to be principal sources of data, but today many additional sources are contributing to the data pool: sensors, social networking sites, web blogs, internet chat rooms, product review websites, online communities, Web pages, email, images, documents, videos and music. This is often topsy-turvy – unstructured– and seems somewhat out of the place in the ordered – structured – world of the past. Data has become a factor of production, according to The Economist's report, almost on par with labour and capital. IDC has predicted that the digital world will be 44 times in 2020 of what it was in 2009, totalling a whopping 35 zeta bytes. EMC has reported the number of customers who are storing a petabytes or more of data to grow from 1,000 to 100,000 within the next 10 years.

There has been a shift in the architecture of data-processing systems today, from the centralized architecture to the distributed architecture. Enterprises face the challenge of processing these huge chunks of data, and have found that none of the existing centralized architectures can efficiently handle this huge volume of data. These are thus utilizing distributed architectures to harness this data. Several solutions to the Big Data problem have emerged which includes the Map Reduce environment championed by Google which is now available open-source in Hadoop. Hadoop's distributed processing, Map Reduce algorithms and overall architecture are a major step towards achieving the promised benefits of Big Data.

Map Reduce & Hadoop are the most widely used models used today for Big Data processing. Hadoop is an open-source

large-scale data processing framework that supports distributed processing of large chunks of data using simple programming models. The Apache Hadoop project consists of the HDFS and Hadoop Map Reduce in addition to other modules. The software is modelled to harvest upon the processing power of clustered computing while managing failures at node level. The Map Reduce software framework which was originally introduced by Google in 2004 is a programming model, which now adopted by Apache Hadoop, consists of splitting the large chunks of data, and 'Map' & 'Reduce' phases (Fig. 1). The Map Reduce framework handles task scheduling, monitoring and failures.

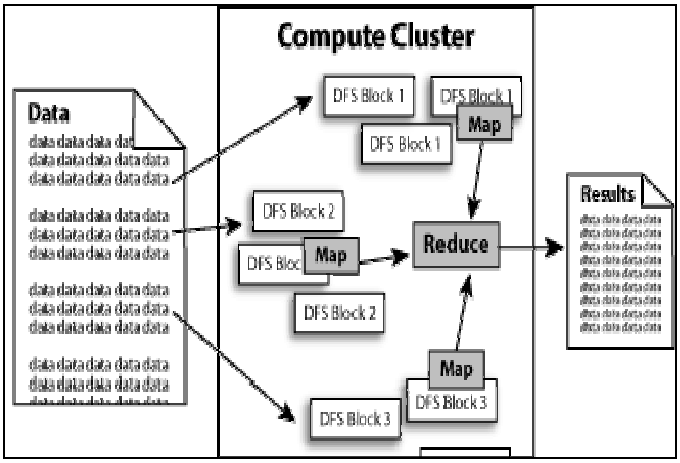

Fig. 1 Map Reduce in Hadoop [5]

2. DISCUSSION

2.1 Introduction to Hadoop

Hadoop is an industrial scale batch processing distributed computing tool. It has the capability to connect computers with multiple processor cores with a scale ranging from hundreds to thousands. Vast volumes of data can be efficiently distributed across clusters of computers using Hadoop.

The Hadoop scale consists of hundreds of gigabytes of data at the least. Hadoop has been built with the capability to manage vast data sets whose size can easily lie between couple of gigabytes to thousands of petabytes. Hadoop provides its solution in the form of a Distributed File System which splits the data and stores it in several different machines. This enables parallel processing of the problem and efficient computation is possible.

The design of Hadoop is such that it can efficiently manage vast quantity of data sets by taking advantage of clustered computing or by connecting hundred of machines with processing power in parallel. Theoretically speaking, a single, powerful thousand CPU machine would be much more expensive than thousands of machines with individual CPUs thus making it an easier investment. Hadoop offers a cost effective solution by tying these smaller and cheaper machines together.

2.2 Distribution of Data

After the data is loaded into clusters in Hadoop it is distributed to all the nodes. The HDFS then splits the data into sets which allow management by individual nodes within the cluster. To handle unavailability of data due to failure, each part is also replicated across the cluster. The data is also re-replicated in response to failure of the system. All these parts of data are easily accessible through a universal namespace, despite the parts being distributed and replicated on multiple machines.

Hadoop follows the policy of "Moving computation to the data" (Fig. 2). As such, data is broken into formats in accordance with the application logic. Hadoop programming framework is record-oriented. A node in the cluster processes a subset of records by a process(s) which are then scheduled using the location information in the file system. The computation is moved to the closest location of the availability of data. Unnecessary data transfers are avoided since much of the information is read from the locally available disk system. Each process on a node processes a subset of data. This strategy greatly enhances performance because of high data locality.

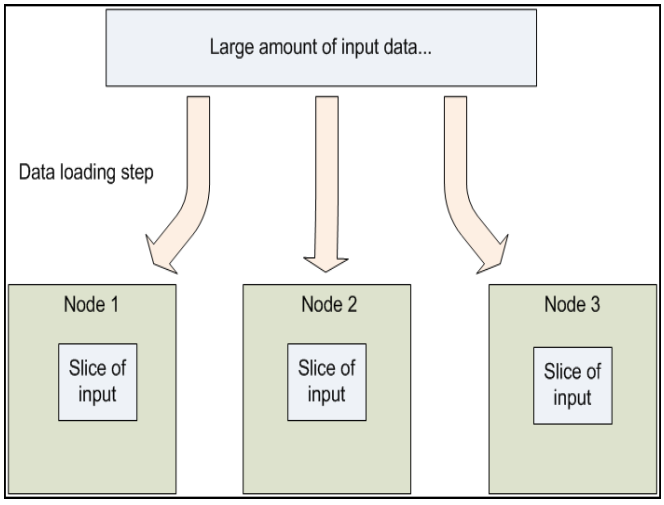

Fig. 2 Moving computation to the data [4]

2.3. Hadoop Distributed File System

HDFS or the Hadoop Distributed File System, is a clustered file management system which aims to hold large datasets (ranging from gigabytes, terabytes to petabytes), and provides high-throughput & quick access to data. The systems stores the files in a redundant manner through a number of machines to ensure that they are fault-tolerant and available to very parallel applications.

The design of HDFS is closely linked to the Google File System or the GFS.

The file system of HDFS is block-structured in which files are broken down into small units of a size that is specified. These units or blocks can then be stored through a loop or clusters of multiple, data storage computing capability. The computing systems in each cluster are called **DataNodes** (Fig. 3). A file can consist multiple blocks, and it is not necessary that they are stored on the same machine as the decision where each block will be stored is randomly selected. As such, locating particular file needs cooperation from multiple machines. If multiple machines are needed in serving a file, then a file could become unavailable even if a single machine in the cluster is lost. HDFS handles this issue by replicating each block across multiple systems which is set to 3 as default.

Block sizes ranging from four to eight kilobytes are used by file systems structured in blocks, mostly. On the other hand, block size of 64MB is used in HDFS by default which is much larger.

It is necessary that this file system stores the metadata reliably. Also, when the file data is being accessed in a WORM model, the structures of the metadata may be modified – even by multiple client systems at the same instance. It is necessary that this information is synchronized. Thus, the whole process is managed by a single system called the **Name Node** which has the metadata of the entire file system. However, because metadata of each file is relatively low, it is possible to store this whole information in main memory of Name Node machine, thus allowing for quicker accessibility.

Opening a file system requires the system to contact the Name Node which is then returned a list of all the locations containing the blocks, together which comprise the file. These locations direct the Data Nodes which hold each block. In this way, the data can be read by the Data Node servers of the clients in parallel. Direct involvement of the Name Node is not there this data transfer, so its overhead is kept to a minimum.

2.4. HDFS Architecture

The HDFS is designed to run on clustered computing platform. It mirrors the already existing file nomenclature in many ways but its differences really make it stands out from existing file systems (Fig. 4). One of the salient features of HDFS is that it is fault-tolerant to a very high degree and cost effective. The system allows for greater and faster access to data of an application which is an advantage for processes that require access to large amount of data. HDFS was designed by Apache Nutch project as an infrastructure extension and is now a core component of the project.

1. Name Node and Data Nodes: HDFS is based on a typical master - slave architecture. An HDFS cluster is made up of a single Name Node and a server acting as a master managing the file access and name space regulations. To simplify the system architecture a single name node exists in a cluster. The Name Node holds & manages whole metadata of HDFS. The design of the systems is such that the data does not flow through the Name Node

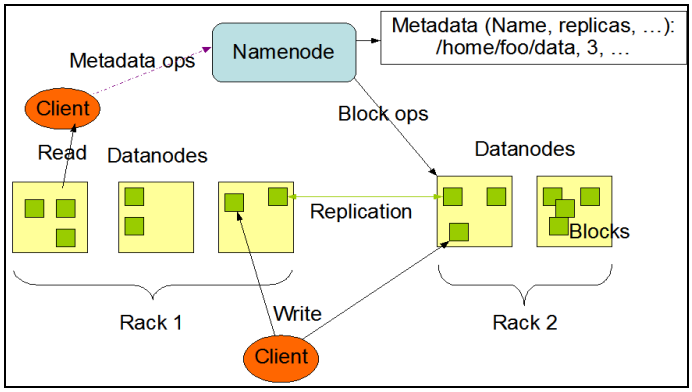

Fig. 3 HDFS Architecture [16]

2. The File System Namespace: HDFS supports an empherical file structure. Directories can be created by user or an application and files are stored inside those directories. The hierarchy of the file namespace is usually like the previously defined file systems. As such files can be created & removed, moved from one directory to another directory or renamed. HDFS has not yet implemented user quotas and access permissions. The Name Node handles the file system namespace. It records alterations and its associated properties. A number can also be specified for replicas of a file by the application which must be maintained by the HDFS which is defined as the replication factor and the information is stored in the Name Node.

3. Data Replication: HDFS is programmed to manage last file stored in large cultures of data mines / structures while ensuring reliability. The way this is managed is by storing files in a sequence of blocks which are the same size, with the last block being an exception. These blocks are then replicated to test fault tolerance in which the size of the block and the replication factors are configurable. An application can then custom specify the number of copies of a file.

Decisions relating to block replications are taken by the Name Node which receives a Heartbeat and Block Report at timed intervals from Data Node in a cluster. In this way it can be ensured that Data Node is functional in the way it is supposed to be.

2.5 Map Reduce

Map Reduce is a tool implemented for managing and processing vast amounts of unstructured data in parallel based

on division of a big work item in smaller independent task units. Programs which are Map Reduces are programmed to manage vast amounts of data in parallel. To achieve this, load shedding is required across multiple machines. The main leverage of MAPREDUCE is the tasks of similar nature are grouped together so that same type of data is placed on the same nodes. Doing this saves the sync overhead which might have been caused if tasks were grouped in a random order. MAPREDUCE data elements are immutable i.e if you change input (key, value) in a mapper then it will not be displayed in the input files. Rather it will be taken care in the next execution with the new output values (key, value).

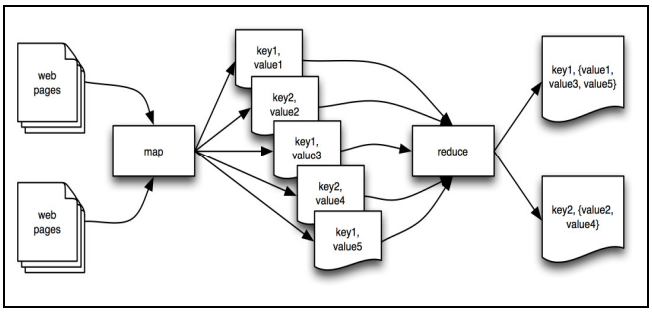

Fig. 4 Mapping & Reducing [24]

1. List Processing: In concept, programs which are map reduced convert an array of data coming in as input into an array of data which is the output. The program goes through this process two times, using two functions which are *mapping* and *reduction*.
2. List based Mapping: In a map reduce context the first execution phase is the MAPPER which takes the data elements as input and generates the corresponding output data elements.**s**
3. List based Reduction: The Reducing part allows us to consolidate the values together. List of values are input to the reducer function from the input list and as an output we receive single output value.

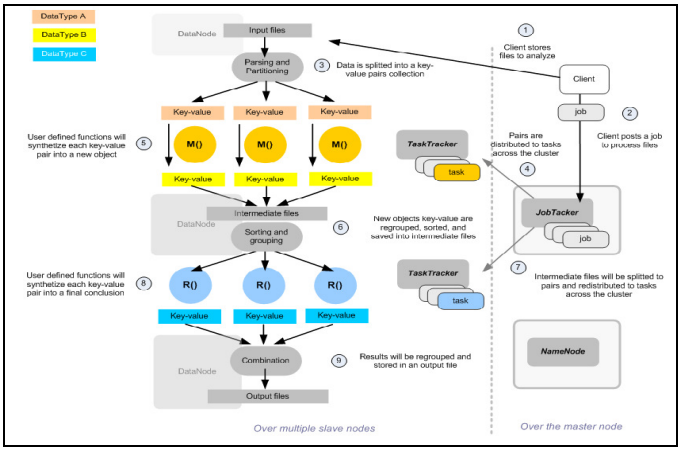

Fig. 5 The Map & Reduce Lifecycle[24]

3. CONCLUSION

Hadoop with its efficient DFS & programming framework based on concept of mapped reduction, is a powerful tool to manage large data sets. With its map-reduce programming paradigms, overall architecture, ecosystem, fault- tolerance techniques and distributed processing, Hadoop offers a complete infrastructure to handle Big Data. Users must leverage the benefits of Big-Data by adopting Hadoop infrastructure for data processing. However, the issues such as lack of flexible resource management, application deployment support, and multiple data source support pose a challenge to Hadoop's adoption. Proper skill training is also needed for achieving large scale data analysis. These challenges must be overcome so that we can tap the full potential of Hadoop data management power.

REFERENCES

[1] Yahoo! Inc, Hadoop Tutorial from Yahoo! Available: http://developer.yahoo.com/hadoop/tutorial/index.html

[2] Jens Dittrich and JorgeArnulfo Quian´eRuiz, " Efficient Big Data processing in Hadoop Mapreduce," *Proceedings of the VLDB Endowment*, Volume 5 Issue 12, August 2012, Pages 2014-2015

[3] Jens Dittrich, Stefan Richter and Stefan Schuh, " Efficient OR Hadoop: Why Not Both?," *Datenbank-Spektrum*, Volume 13, Issue 1, pp 17-22

[4] Humbetov, S, "Data-Intensive Computing with Map-reduce and Hadoop," in *Proc. 2012 Application of Information and Communication Technologies (AICT),* IEEE,*6th International Conference pp. 5*

[5] Hadoop Tutorial, Apache Software Foundation, 2014, Available: http://hadoop.apache.org/

[6] Sherif Sakr, Anna Liu and Ayman G. Fayoumi, " The family of mapreduce and large-scale data processing systems," *ACM Computing Surveys*, Volume 46 Issue 1, October 2013, Article No. 11

[7] Aditya B. Patel, Manashvi Birla and Ushma Nair, " Addressing Big Data Problem Using Hadoop and Map Reduce," in *Proc. 2012 Nirma University International Conference On Engineering*, pp. 1-5.

[8] Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Rasin, A., and Silberschatz, A. 2009. HadoopDB: An architectural hybrid of mapreduce and dbms technologies for analytical workloads. Proc. VLDB Endow. 2, 1,922–933.

[9] Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Rasin, A., and Silberschatz, A. 2010. HadoopDB in action: Building real world applications. In Proceedings of the 36th ACM SIGMOD International Conference on Management of Data (SIGMOD'10).

[10] Parallel Data Processing with MapReduce: A Survey: www.cs.arizona.edu/~bkmoon/papers/sigmodrec11.pdf

[11] Jyoti Nandimath, Ankur Patil, Ekata Banerjee,Pratima Kakade and Saumitra Vaidya, " Big Data Analysis Using Apache Hadoop," IEEE IRI 2013, August 14-16, 2013, San Francisco, California, USA

[12] MapReduce: Simplified Data Processing on Large Clusters. Available at http://labs.google.com/papers/mapreduceosdi04.pdf

[13] Stephen Kaisler, Frank Armour, J. Alberto Espinosa, William Money, “Big Data: Issues and Challenges Moving Forward”, IEEE, 46th Hawaii International Conference on System Sciences, 2013.

[14] Sam Madden, “From Databases to Big Data”, IEEE, Internet Computing, May-June 2012.

[15] Yuri Demchenko, Zhiming Zhao, Paola Grosso, Adianto Wibisono, Cees de Laat, “Addressing Big Data Challenges for Scientific Data Infrastructure”, IEEE, 4th International Conference on Cloud Computing Technology and Science, 2012.

[16] HDFS Architecture Guide [Online] Available: http://hadoop.apache.org/docs/hdfs/current/hdfs_design

[17] Levy E. and Silberschatz A., "Distributed FileSystems:Concepts and Examples"

[18] Apache Hadoop - Petabytes and Terawatts [Online].Available:http://www.youtube.com/watch?v=SS27FhYWfU& feature=related

[19] Jeffrey Dean and Sanjay Ghemawat. "Mapreduce: simplified data processing on large clusters", Commun. ACM, 51(1):107–113, 2008.

[20] Sanchita Kadambari, Praveen Kumar and Seema Rawat " A comprehensive study on Big Data and its future opportunities," in *Proc. 2014 Fourth International Conference on Advanced Computing & Communication Technologies, pp. 277-281*

[21] Jeffrey Dean and Sanjay Ghemawat, " MapReduce: a flexible data processing tool," *Communications of the ACM*, Volume 53 Issue 1, January 2010, Pages 72-77

[22] “Big Data: The next frontier for innovation, competition, and productivity”, McKinsey Global Institute, May 2011, p. 11: http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation.

[23] Securing Big Data : Architectural Issues Available at https://securosis.com/blog/securing-big-data-architectural-issues

[24] Hadoop’s Programming Model Available: http://adcalves.wordpress.com/2010/12/12/a-hadoop-primer/

[25] The Hadooper in me Available: http://hadooper.blogspot.in/

[26] A. Abouzeid, K. Bajda-Pawlikowski, D. J. Abadi,A. Rasin, and A. Silberschatz. HadoopDB: an architecturalhybrid of MapReduce and DBMS technologiesfor analytical workloads. Proceedings of the VLDB En-dowment (PVLDB), 2(1):922{933, 2009.

[27] D. Agrawal, S. Das, and A. E. Abbadi. Big Data and cloud computing: current state and future opportunities.In Proceedings of International Conference on Extending Database Technology (EDBT), pages 530-533,2011.

[28] D. Borthakur, J. Gray, J. S. Sarma, K. Muthukkaruppan,N. Spiegelberg, H. Kuang, K. Ranganathan,D. Molkov, A. Menon, S. Rash, R. Schmidt, and A. S.Aiyer. Apache Hadoop goes realtime at Facebook. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), pages 1071-1080, 2011.

[29] K. Shim. MapReduce algorithms for Big Data analysis.Proceedings of the VLDB Endowment (PVLDB),5(12):2016-2017, 2012.