Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 48 (2015) 45 – 50

Procedia Computer Science

International Conference on Intelligent Computing, Communication & Convergence

(ICCC-2015)

Conference Organized by Interscience Institute of Management and Technology,

Bhubaneswar, Odisha, India

Hadoop, MapReduce and HDFS: A Developers Perspective

Mohd Rehan Ghazi[a], Durgaprasad Gangodkar[a]

[a]*Graphic Era University, 566/6 Bell Road, Clement town, Dehradun – 248002, India*

Abstract

The applications running on Hadoop clusters are increasing day by day. This is due to the fact that organizations have found a simple and efficient model that works well in distributed environment. The model is built to work efficiently on thousands of machines and massive data sets using commodity hardware. HDFS and MapReduce is a scalable and fault-tolerant model that hides all the complexities for Big Data analytics. Since Hadoop is becoming increasingly popular, understanding technical details becomes essential. This fact inspired us to explore Hadoop and its components in-depth. The process of analysing, examining and processing huge amount of unstructured data to extract required information has been a challenge. In this paper we discuss Hadoop and its components in detail which comprise of MapReduce and Hadoop Distributed File System (HDFS). MapReduce engine uses JobTracker and TaskTracker that handle monitoring and execution of job. HDFS a distributed file-system which comprise of NameNode, DataNode and Secondary NameNode for efficient handling of distributed storage purpose. The details provided can be used for developing large scale distributed applications that can exploit computational power of multiple nodes for data and compute intensive applications.

Peer-review under responsibility of scientific committee of International Conference on Computer, Communication and Convergence (ICCC 2015)

Keywords: Hadoop ; HDFS ; MapReduc e; JobTracker ; TaskTracke r; NameNode ; DataNode.

1. Introduction

Nowadays data is being generated with a very high rate from different fields like business, scientific data, e-mails, blogs, etc. To analyse and process this huge amount of data and to extract meaningful information for users there is

Peer-review under responsibility of scientific committee of International Conference on Computer, Communication and Convergence (ICCC 2015)
doi:10.1016/j.procs.2015.04.108

a need of deploying data intensive application and storage clusters[1]. Requirements for this type of application are fault tolerance; parallel processing, data-distribution, load balancing, scalability and highly availability. To deal with such type of problems Google introduced the MapReduce programming model[2]. Apache Hadoop is an open source implementation of MapReduce system[3].

Hadoop is the most popular and open source implementation of MapReduce programming model. Apache Hadoop is a software framework for reliable, scalable, parallel and distributed computing. In lieu of relying on costly hardware and expensive systems for processing and storing data, Apache Hadoop empowers parallel processing on Big Data on commodity hardware[4]. HDFS and MapReduce programming standard grant itself to these Big Data intensive analytic jobs because of its scale-out architecture and its ability to process data in parallel manner in multi-node clusters. MapReduce, and its existing open-source project called Hadoop, enables parallel processing of huge amount of data and automatic partition of data, data distribution, fault tolerance and load balancing management which finally yields reliable and scalable computing. The rapid and high growth rate of data first posed challenges on big companies like Facebook, Google, Amazon and Yahoo. These companies need to execute and carry out terabytes and petabytes of data on daily basis to deduce demands and queries of their users. Existing and traditional tools and application becomes deficient to process large amount of data. Hadoop explained and answered the problem of handling and processing such terabytes and petabytes of data[5]. Many enterprises, industries and universities works on parallel and distributed computing but for neophytes of Big Data and its processing and analysis is not an easy task and it requires much effort to manage due to the lack of experience and insufficient money to invest in super computers and servers. For such neophytes it is important to utilize limited resources in effective and efficient manner.

In this paper the section II provide the explanation and relation of cloud computing with hadoop and why and how big organization exploiting hadoop along with cloud computing. Section III comprised of Hadoop Architecture which explains all the daemons of Hadoop including JobTracker, TaskTracker, NameNode, DataNode and Secondary NameNode in detail followed by conclusion.

2. Cloud computing with hadoop

Enterprises like Google, Facebook other Internet colossal organizations process user demands, queries and workloads. Millions and billions of queries and requests are served every hour on Internet. Therefore, migrating towards cloud computing to take advantage of reduced costs and improved performance is becoming need of an hour[6]. The key is to select the applications and technology that is best suited for organization's mission, infrastructure, and long-term service. Apache Hadoop offers a broad selection of effective cloud deployable solutions to assist organizations in achieving its goal and in successfully migrating to the cloud.

These organizations depend on Linux servers and cloud computing for getting the adequate performance in terms of scalability and availability[7]. The pliability of Linux merged with smooth and consistent scalability of cloud environment makes it capable of supplying the ideal framework for analysing and processing Big Data, while eliminating the need for costly hardware and software. Hadoop is seen as a preferred choice in open source cloud computing community for providing an efficient platform for Big Data processing. It has become obvious that Apache Hadoop in cloud computing is now an interesting topic because cloud computing is regarded as the next milestone of the IT industry[8].

3. Hadoop Architecture

Apache Hadoop comprise of five different daemons and each of these daemons run its own JVM- 1) NameNode, 2) DataNode, 3) Secondary NameNode, 4) JobTracker and 5)TaskTracker[9]. Daemons which stores data and metadata i.e., NameNode and DataNode, comes under HDFS layer and JobTracker and TaskTracker, as shown in Fig. 1, which keeps track and actually executes the job, comes under MapReduce layer. Hadoop cluster comprised of multiple slave nodes and a master node. The master node runs the master daemons for each layer i.e., NameNode for the HDFS storage layer, and JobTracker for the MapReduce processing layer. Rest of the machines will run the "slave" daemons: DataNode for the HDFS layer and TaskTracker for MapReduce layer.

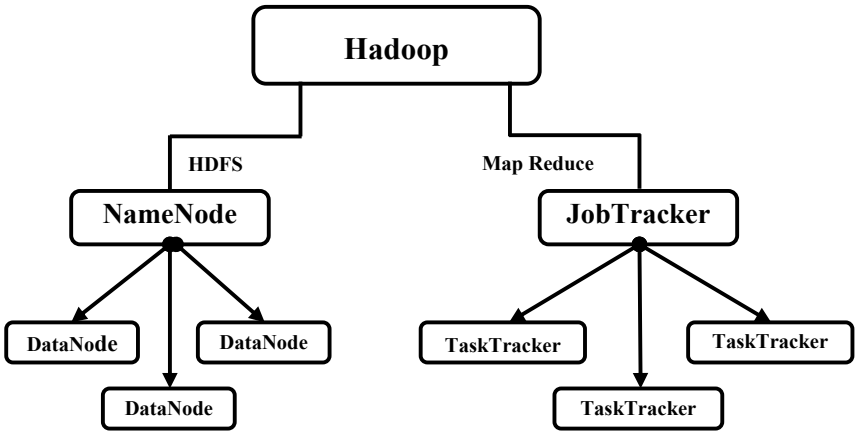

Fig 1: Hadoop Daemons

Master node can also play a role of slave. Thus, in addition to master daemons, master node can run the slave daemons as well. Fundamentally, the daemons running on master node take responsibility for coordinating and managing the slave daemons on all nodes which carryout work for data storage and processing[10,11].

Hadoop core architecture comprise of –

3.1. HDFS (Hadoop Distributed File System) – HDFS is a self-healing, distributed file system that provides reliable, scalable and fault tolerant data storage on commodity hardware[12,13]It works closely with MapReduce by distributing storage and computation across large clusters by combining storage resources that can scale depending upon requests and queries while remaining inexpensive and in budget[14,15]. HDFS accepts data in any format like text, images, videos, etc regardless of architecture and automatically optimizes for high bandwidth streaming.

The foremost advantage of HDFS is fault tolerance. By provisioning fast data transfer between the nodes and enabling Hadoop to continue to provide service even in event of node failures decreases the risk of catastrophic failure[16]. HDFS is also capable of providing scale-out storage solution for Hadoop.

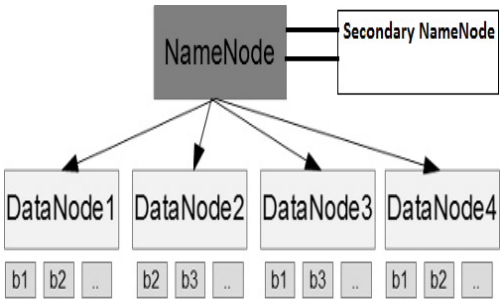

Fig 2: HDFS Architecture

HDFS exploits master/slave architecture with NameNode daemon and secondary NameNode running on master node and DataNode daemon running on every single salve node as shown in Fig. 2. HDFS storage layer comprised of three daemons –

a) NameNode: A single NameNode daemon runs on master node. NameNode stores and manages metadata about the file system in a file named *fsimage*. This metadata is cached in main memory to provide faster access to the clients on read/write requests. The NameNode controls the slave DataNode daemons to execute the I/O tasks.

The NameNode manages and controls how files are broken down into blocks, identifies which slave node should store these blocks, along with the overall condition and fitness of the distributed file system. The tasks carried out by the NameNode are memory and I/O intensive.

b) DataNodes: Hadoop cluster comprise of DataNode daemon that runs on every slave node. DataNodes are primary storage elements of HDFS that store data blocks and service read/write requests on files stored on HDFS. These are controlled by NameNode. Blocks stored in DataNodes are replicated as per the configuration to provide reliability and high availability. These replicated blocks are distributed across the cluster to provide rapid computation.

c) Secondary NameNode: Secondary NameNode is not a backup for the NameNode. The Secondary NameNode's job is to periodically read the file system, log the changes and apply them into the *fsimage* file. This helps in updating NameNode to start up faster next time as shown in Fig. 3.

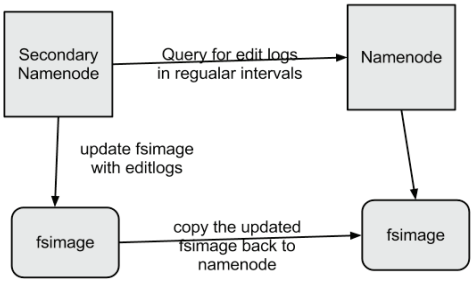

Fig 3: Secondary NameNode task

3.2. MapReduce: MapReduce is programming model or a software framework used in Apache Hadoop. Hadoop MapReduce is provided for writing applications which process and analyze large data sets in parallel on large multinode clusters of commodity hardware in a scalable, reliable and fault tolerant manner. Data analysis and processing uses two different steps namely, Map phase and Reduce phase[17].

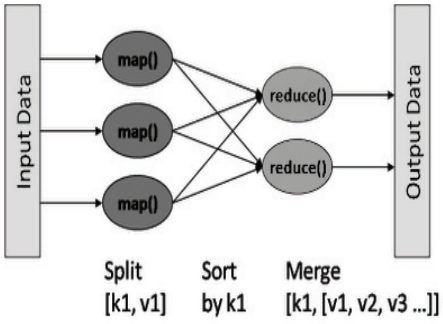

Fig 4: Map and Reduce Phase[18]

A MapReduce job generally breaks and divides the input data into chunks which are first processed by "Map phase" in parallel and then by "Reduce phase" [19, 20, 21]. Hadoop framework sorts out the output of the Map phase

which are then given as an input to Reduce phase to initiate parallel reduce tasks as shown in Fig. 4. These input and output files are stored in file system. By default, the MapReduce framework gets input datasets from HDFS[22, 23]. It is not necessary that Map and Reduce tasks proceed in a sequential manner i.e., reduce tasks can begin as soon as any of the Map task completes its assigned work. It is also not necessary that all Map tasks completes before any reduce tasks starts working. MapReduce works on key-value pairs. Conceptually, a MapReduce task takes input data set as key-value pair and gives output in the form of key-value pair only by processing input data sets through MapReduce phases. Output generated by the Map phase is called intermediate results which are given as an input to reduce phase as shown in Fig. 5.

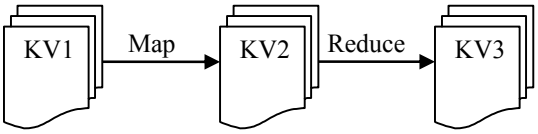

Fig 5: MapReduce key-value pairs

Similar to HDFS, MapReduce also exploits master/slave architecture in which JobTracker daemon runs on master node and TaskTracker daemon runs on each salve node as shown in Fig. 6. MapReduce processing layer comprised of two daemons –

a) JobTracker: The JobTracker service runs on master node and monitors MapReduce tasks executed by TaskTracker on salve nodes. User in interaction with the Master node submits the job to the JobTracker. JobTracker then asks NameNode for the actual location of data in HDFS to be processed. JobTracker locates TaskTracker on slave nodes and submits the jobs to TaskTracker on slave nodes. The TaskTracker sends heartbeat message back to JobTracker periodically to ensure that TaskTracker on a particular slave node is alive and executing its allotted job. If the heartbeat message is not received within a stipulated time then TaskTracker on a particular salve node is considered to be non-functional and the assigned job is scheduled on another TaskTracker. JobTracker and TaskTracker are known as the MapReduce engine. JobTracker is a single point-of-failure for Hadoop MapReduce service, if it goes down all executing jobs will be stopped.

b) TaskTracker: A TaskTracker daemon runs on slave nodes of a cluster. It accepts jobs from JobTracker and executes MapReduce operations. Each TaskTracker has a finite number of “task slots” based on the ability of a node. The heartbeat protocol permits JobTracker to know how many “task slots” are available in TaskTracker on a slave node. It is the task of JobTracker to allocate appropriate jobs to the particular TaskTracker depending upon how many free task slots are available.

TaskTracker governs the execution of every MapReduce operation on each slave node. Although, there is only one TaskTracker per slave node, each TaskTracker can start multiple JVM’s to execute MapReduce operation in parallel. TaskTracker from every slave node sends heartbeat message to JobTracker, master node, periodically to ensure JobTracker that TaskTracker is still alive.

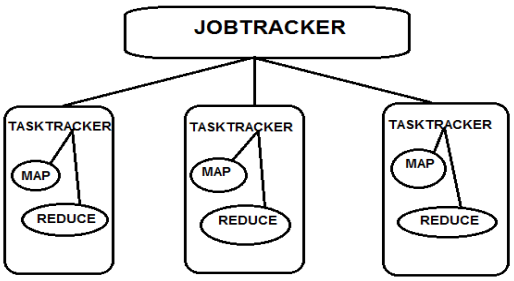

Fig 6: JobTracker and TaskTracker

4. Conclusion

Hadoop MapReduce programming paradigm and HDFS are increasingly being used for processing large and unstructured data sets. Hadoop enables interacting with the MapReduce programming model while hiding the complexity of deploying, configuring and running the software components in the public or private cloud. Hadoop enables users to create cluster of commodity servers. MapReduce has been modelled as an independent platform-as-a-service layer suitable for different requirement by cloud providers. It also enables users to understand the data processing and analysing.

References

1. C.A. Hansen, "Optimizing Hadoop for the cluster", *Institute for Computer Science, University of Troms, Norway.* [online].Available:http://oss.csie.fju.edu.tw/~tzu98/optimizing %20Hadoop%20for%20the%20cluster.pdf
2. S.Ghemawat, H.Gobioff and S.T.Leung, " The google file system". in *Proceedings of the 19th ACM symposium on operating systems principles*, Lake George, New York, USA, Feb. 10, 2003, vol. 37, No. 5, pp. 29-43.
3. C. Lam, "Introducing Hadoop", in *Hadoop in Action*, MANNING, 2011.
4. P.D. Londhe, S.S. Kumbhar, R.S. Sul, and A.J. Khadse, "Processing big data using hadoop framework", in *Proceedings of 4th SARC-IRF International Conference,* New Delhi, India, Apr. 27, 2014, pp. 72-75.
5. Kenneth Wottrich, and T. Bressoud, "The performance characteristics of mapreduce applications on scalable clusters", in *Proceedings of the Midstates Conference on Undergraduate Research in Computer Science and Mathematics (MCURCSM),* Denison University, Granville, USA, Nov. 2011.
6. S. Loughran, J.M.A. Calero, A. Farrell, J. Kirschnick, and J.Guijarro, "Dynamic deployment of mapreduce architecture in the cloud."*IEEE Internet Computing*, vol. 16, no. 6, pp. 40-50, Dec. 2012.
7. Z. Zhang, L. Cherkasova, and B.T. Loo, "Performance modeling of mapreduce jobs in heterogeneous cloud environments", in *Proceedings of the 6th IEEE International Conference on Cloud Computing,* IEEE Computer Society, Santa Clara Marriott, CA, USA, Jun. 27-Jul. 2, 2013, pp. 839-846.
8. B.T. Rao, N.V. Sridevi, V.K. Reddy, and L.S.S. Reddy, "Performance issues of heterogeneous hadoop clusters in cloud computing", *Global Journal of Computer Science and Technology,* vol. 11, no. 8, May, 2011.
9. Z. Benslimane, Q. Liu, and Z. Hongming, "Predicting hadoop parameters", in *Proceedings of the Second International Conference on Advances in Electronics and Electrical Engineering (AEEE),* Seek digital library, IRED Headquarters, Santa Barbara, California USA, Apr. 6, 2013, pp. 63-67.
10. T. White, "MapReduce and the hadoop distributed file system", *in Hadoop: The definitive guide*, 1st edition, O'Reilly Media, Inc., Yahoo press, 2012.
11. A.Elsayed, O. Ismail, and M.E. El-Sharkawi, "MapReduce: state-of-the-art and research directions", *International Journal of Computer and Electrical Engineering*, vol. 6, no. 1, pp. 34-39, Feb. 2014
12. D. Borthakur, "The hadoop distributed file system: architecture and design," *Hadoop Project Website* [online]. Available: http://hadoop.apache.org/ core/docs/current/hdfs design.pdf
13. J.Venner and S.Cyrus, "The Mapreduce", in *Pro Hadoop*, vol. 1, New York, APRESS, 2009.
14. S. Sur, H. Wang, J. Huang, X. Ouyang, and D.K. Panda, "Can high-performance interconnects benefit hadoop distributed file system," in *Workshop on Micro Architectural Support for Virtualization, Data Center Computing, and Clouds (MASVDC). Held in Conjunction with the 43rd IEEE/ACM International Symposium on Microarchitecture (MICRO-43),* Atlanta, GA, USA, Dec. 2010.
15. J. Shafer, S. Rixner, and A.L. Cox, "The hadoop distributed filesystem: balancing portability and performance", in *IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS),* White Plains, NY, Mar. 28-30, 2010, pp. 122-133.
16. F. Faghri, S. Bazarbayev, M. Overholt, R. Farivar, R. H. Campbell and W. H. Sanders, "Failure scenario as a service (FSaaS) for hadoop clusters," in *Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management ACM*, Beijing, China, Oct. 1-11, 2012, pp. 5
17. J.Dean and S.Ghemawat. "Mapreduce: simplified data processing on large clusters", *In OSDI'04: Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation,* USENIX Association, Berkeley, CA, USA, Nov. 4, 2004, vol. 51, no. 1, pp. 107–113.
18. J. Lin and C. Dyer, "Data-intensive text processing with mapreduce", in *Synthesis Lectures on Human Language Technologies*, vol. 3, no. 1, pp.1-177, 2010.
19. J. Ekanayake, S. Pallickara, and G. Fox, "Mapreduce for data intensive scientific analyses", in *IEEE 4th International Conference on eScience*, Indianapolis, Indiana, USA, Dec. 7-12, 2008, pp. 277-284.
20. Gray, and T.C. Bressoud, "Towards a mapreduce application performance model", in *Proceedings of the Midstates Conference on Undergraduate Research in Computer Science and Mathematics (*MCURCSM), Ohio Wesleyan University, USA, Nov.17, 2012.
21. A. Verma, N. Zea, B. Cho, I. Gupta and R. H. Campbell, "Breaking the mapreduce stage barrier", in *Proceeding of IEEE International Conference on Cluster Computing (CLUSTER),,* Chicago, USA, Sep. 23-27, 2013, vol. 16, no. 1, pp. 191-206.
22. C. Tian, H. Zhou, Y. He, and L. Zha, "A dynamic mapreduce scheduler for heterogeneous workloads." in *8th International Conference on Grid and Cooperative Computing (GCC)*, Lanzhou, China, Aug. 27-29, 2009, pp. 218-224.
23. C. Lam, "Writing basic mapreduce programs", in *Hadoop in Action*, 1st ed., MANNING, 2011.