APACHE HADOOP

[1]*K. Anuradha* , *M. Savariselvi*[2]

ABSTRACT

Hadoop is the popular open source implementation of MapReduce, a powerful tool designed for deep analysis and transformation of very large data sets. Hadoop enables you to explore complex data, using custom analyses tailored to your information and questions. Hadoop is the system that allows unstructured data to be distributed across hundreds or thousands of machines forming shared nothing clusters, and the execution of Map/Reduce routines to run on the data in that cluster. Hadoop has its own filesystem which replicates data to multiple nodes to ensure if one node holding data goes down, there are at least 2 other nodes from which to retrieve that piece of information. This protects the data availability from node failure, something which is critical when there are many nodes in a cluster (aka RAID at a server level). Hadoop has its origins in Apache Nutch, an open source web search engine, itself a part of the Lucene project. Building a web search engine from scratch was an ambitious goal, for not only is the software required to crawl and index websites complex to write, but it is also a challenge to run without a dedicated operations team, since there are so many moving parts.

This paper does detailed study on Hadoop architecture and component working. Later focus on how data replicas are managed in Hadoop distributed file system for better performance and high data availability for highly parallel distributed Hadoop Applications. This paper also takes in account the different failure which will affect the Hadoop system and various failover mechanisms for handling those failures.

1. INTRODUCTION:

Dealing with "Big Data" requires – an in expensive, reliable storage and a new tool for analyzing structured and unstructured data. Today, we're surrounded by big data. People upload videos, take pictures on their cell phones, text friends, update their Facebook status, leave comments around the web, click on ads, and so forth. Machines, too, are generating and keeping more and more data. We live in the data age. It's not easy to measure the total volume of data stored electronically, but an IDC estimate put the size of the "digital universe" at 0.18 zettabytes in 2006, and is forecasting a tenfold growth by 2011 to 1.8 zettabytes. A zettabyte is 10^21 bytes, or equivalently one million petabytes, or one billion terabytes. That's roughly the same order of magnitude as one disk drive for every person in the world. Thus

[1]Associate Professor/MCA
Karpagam College of Engg., Coimbatore.
anu_kce@yahoo.com

[2]MCA 3rd YEAR
Karpagam College of Engg., Coimbatore.
selvi.ms117@gmail.com

big data is a term applied to data sets whose size is beyond the ability of commonly used software tools to capture, manage, and process the data within a tolerable elapsed time. Big data sizes are a constantly moving target currently ranging from a few dozen terabytes to many petabytes of data in a single data set.

The exponential growth of data presented challenges to cutting-edge businesses such as Google, Yahoo, Amazon, and Microsoft. They needed to go through terabytes and petabytes of data to figure out which websites were popular, what books were in demand, and what kinds of ads appealed to people. Existing tools were becoming inadequate to process such large data sets. Google was the first to publicize MapReduce a system they had used to scale their data Processing needs. This system aroused a lot of interest because many other businesses were facing similar scaling challenges, and it wasn't feasible for everyone to reinvent their own proprietary tool. Doug Cutting saw an opportunity and led the charge to develop an open source version of this MapReduce system called Hadoop. Soon after, Yahoo, rallied around to support this effort. Today, Hadoop is a core part of the computing infrastructure for many web companies, such as Yahoo, Facebook, LinkedIn, and Twitter. Many more traditional businesses, such as media and telecom, are beginning to adopt this system too. Thus Hadoop is an open source framework for writing and running distributed applications that process large amounts of data. Hadoop distribute and parallelize data processing across many nodes in a compute cluster, speeding up large computations and hiding I/O latency through increased concurrency. It is well suited for large data processing like searching and indexing in huge data set.

Hadoop includes Hadoop Distributed File System (HDFS) and Map Reduce. It is not possible for storing large amount of data on a single node, therefore Hadoop use a new file system called HDFS which split data into many smaller parts and distribute each part redundantly across multiple nodes. MapReduce is a software framework for the analysis and transformation of very large data sets. Hadoop uses Map Reduce function for distributed computation. MapReduce programs are inherently parallel. Hadoop take advantage of data distribution by pushing the work involved in analysis to many different servers. Each server runs the analysis on its own block from the file. Results are combined in to singleresult after analysing each piece. MapReduce framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

1.1. What is Hadoop?

Consider the example of Facebook, Facebook data has grown up to 15TB/day by 2011 and in future shall produce data of a much higher magnitude. They have many web servers and huge MySQL (profile, friends etc.) servers to hold the user data.

Now to run various reports on these huge data For example Ratio of men vs. women users for a period, No of users who commented on a particular day. The solution for this requirement they had scripts written in python which uses ETL processes. But as the size

of data increased to this extent these scripts did not work. Hence their main aim at this point of time was to handle data warehousing and their home ground solutions were not working. This is when Hadoop came into the picture.

Formally speaking, Hadoop is an open source framework for writing and running distributed applications that process large amounts of data. Distributed computing is a wide and varied field, but the key distinctions of Hadoop are that it is

% **Accessible**—Hadoop runs on large clusters of commodity machines or on **cloud computing** services such as Amazon's Elastic Compute Cloud (EC2)

% **Robust**—because it is intended to run on commodity hardware, Hadoop is architected with the assumption of frequent hardware malfunctions. It can gracefully handle most such failures.

% **Scalable**—Hadoop scales linearly to handle larger data by adding more nodes to the cluster.

% **Simple**—Hadoop allows users to quickly write efficient parallel code.

Hadoop was created by **Doug Cutting**, the creator of Apache Lucene, the widely used text search library. Hadoop has its origins in Apache Nutch, an open source web search engine, itself a part of the Lucene project. Hadoop's accessibility and simplicity give it an edge over writing and running large distributed programs. On the other hand, its robustness and scalability make it suitable for even the most demanding jobs at Yahoo and Facebook. These features make Hadoop popular in both academia and industry.

2. Hadoop System Architecture:

The Hadoop architecture system consists of following components:

HBase: HBase is an open source, non-relational, distributed database modelled after Google's Big Table and written in Java. A distributed, column oriented database. HBase uses HDFS for its underlying Storage, and supports both batch-style computations using MapReduce and point queries .HBase features compression, in-memory operation, and Bloom filters on a per-column basis as outlined in t he original Big Table paper. Tables in HBase can serve as the input and output for MapReduce jobs run in Hadoop. HBase is not a direct replacement for a classic SQL database, although recently its performance has improved, and it is now serving several data-driven websites, including Facebook's Messaging Platform.

HDFS: A distributed filesystem that runs on large clusters of commodity machines.

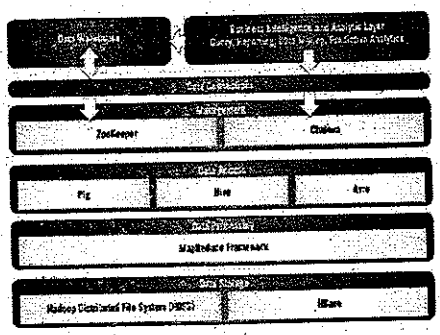


Figure 1. Hadoop System Architecture

MapReduce: MapReduce is a functional programming paradigm that is well suited to handling parallel processing of huge data sets distributed across a large number of computers, or in other words, MapReduce is the application paradigm supported by Hadoop and the infrastructure presented in this article. MapReduce, as its name implies, works in two steps:

Avro: is the serialization framework created by Doug Cutting, the creator of Hadoop. With Avro we can store data and read it easily with various programming languages. It is optimized to minimize the disk space needed by our data and it is flexible – after adding or removing fields to our data we can still keep reading files previous to the change.

Hive: A distributed data warehouse. Hive manages data stored in HDFS and provides a query language based on SQL for querying the data. Hive looks very much like traditional database code with SQL access. However, because Hive is based on Hadoop and MapReduce operations, there are several key differences. The first is that Hadoop is intended for long sequential scans, and because Hive is based on Hadoop, you can expect queries to have a very high latency (many minutes). This means that Hive would not be appropriate for applications that need very fast response times, as you would expect with a database such as DB2. Finally, Hive is read-based and therefore not appropriate for transaction processing that typically involves a high percentage of write operations.

Pig: A data flow language and execution environment for exploring very large datasets. Pig runs on HDFS and MapReduce clusters. **Pig** was initially developed at Yahoo to allow people using Hadoop to focus more on analyzing large data sets and spend less time having to write mapper and reducer programs. Pig is made up of two components: the first is the language itself, which is called **Pig Latin** and the second is a runtime environment where Pig Latin programs are executed.

Chukwa: Chukwa is a data collection and Analysis Framework that works with Hadoop to process and analyze the huge logs generated. It is built on top of the Hadoop Distributed File System (HDFS) and Map Reduce Framework. It is highly flexible tool that makes Log analysis, processing and monitoring easier especially while handling Distributed File Systems like Hadoop.

ZooKeepe: Zookeeper is an open source Apache project that this information in local log files. A very large Hadoop cluster can be supported by multiple ZooKeeper servers (in this case, a master server synchronizes the top-level servers). Each client machine communicates with one of the ZooKeeper servers to retrieve and update its synchronization information. Within ZooKeeper, an application can create what is called a znode. The znode can be updated by any node in the cluster, and any node in the cluster can register to be informed of changes to that znode. Using this znode infrastructure, applications can synchronize their tasks across the distributed cluster by updating their status in a

ZooKeeper znode, which would then inform the rest of the cluster of a specific node's status change. This cluster-wide status centralization service is essential for management and serialization tasks across a large distributed set of servers.

Sqoop: Using Hadoop for analytics and data processing requires loading data into clusters and processing it in conjunction with other data that often resides in production databases across the enterprise. Loading bulk data into Hadoop from production systems or accessing it from map reduce applications running on large clusters can be a challenging task. This is where Apache Sqoop fits in. Sqoop allows easy import and export of data from structured data stores such as relational databases, enterprise data ware houseses. What happens underneath the covers when you run Sqoop is very straight forward. The dataset being transferred is sliced up into different partitions and a map-only job is launched with individual mappers responsible for transferring a slice of this dataset. Each record of the data is handled in a type safe manner since Sqoop uses the database metadata to infer the data types.

3. Conclusion

• The Key Benefits of Apache Hadoop:

– Agility/Flexibility (Quickest Time to Insight).

– Complex Data Processing (Any Language, Any Problem).

– Scalability of Storage/Compute (Freedom to Grow).

– Economical Storage (Keep All Your Data Alive Forever).

• The Key Systems for Apache Hadoop are:

– Hadoop Distributed File System: self-healing high bandwidth

– Clustered storage.

– MapReduce: distributed fault-tolerant resource

– Management coupled with scalable data processing.

REFERENCES

1. Apache Hadoop. http://hadoop.apache.org/
2. P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. *"PVFS: Aparallel file system for Linux clusters,"* in Proc. of 4th Annual Linux Showcase and Conference, 2000, pp. 317–327.
3. J. Dean, S. Ghemawat, *"MapReduce: Simplified Data Processing on Large Clusters,"* In Proc. of the 6th Symposium on Operating Systems Design and Implementation, San Francisco CA, Dec. 2004.
4. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C.Olston, B. Reed, S. Srinivasan, U. Srivastava. *"Building a High-Level Dataflow System on top of MapReduce: The Pig Experience,"* In Proc. of Very Large Data Bases, vol 2 no. 2, 2009, pp. 1414–1425
5. S. Ghemawat, H. Gobioff, S. Leung. *"The Google file system,"* In Proc. of ACM

Symposium on Operating Systems Principles, Lake George,NY, Oct 2003, pp 29–43.

6. F. P. Junqueira, B. C. Reed. *"The life and times of a zookeeper,"* In Proc. of the 28th ACM Symposium on Principles of Distributed Computing, Calgary, AB, Canada, August 10–12, 2009.

7. Lustre File System. http://www.lustre.org

8. M. K. McKusick, S. Quinlan. *"GFS: Evolution on Fast-forward,"* ACM Queue, vol. 7, no. 7, New York, NY. August 2009.

9. O. O'Malley, A. C. Murthy. Hadoop Sorts a Petabyte in 16.25 Hours and a Terabyte in 62 Seconds. May 2009.

10. R. Pike, D. Presotto, K. Thompson, H. Trickey, P. Winterbottom, *"Use of Name Spaces in Plan9,"* Operating Systems Review, 27(2), April 1993, pages 72–76.

11. S. Radia, *"Naming Policies in the spring system,"* In Proc. of 1st IEEE Workshop on Services in Distributed and Networked Environments, June 1994, pp. 164–171.

12. S. Radia, J. Pachl, *"The Per-Process View of Naming and Remote Execution,"* IEEE Parallel and Distributed Technology, vol. 1, no. 3, August 1993, pp. 71–80.

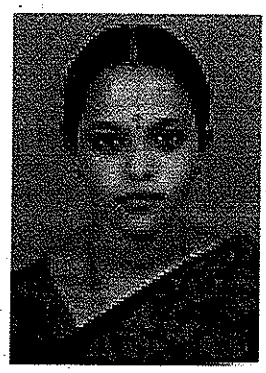
13. K. V. Shvachko, *"HDFS Scalability: The limits to growth,"* login: April 2010, pp. 6–16.

14. W. Tantisiriroj, S. Patil, G. Gibson. *"Data-intensive file systems for Internet services: A rose by any other name ..."* Technical Report CMUPDL- 08-114, Parallel Data Laboratory, Carnegie Mellon University, Pittsburgh, PA, October 2008.

15. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy, *"Hive – A Warehousing Solution Over a Map- Reduce Framework,"* In Proc. of Very Large Data Bases, vol. 2 no. 2, August 2009, pp. 1626-1629.

16. J. Venner, Pro Hadoop. Apress, June 22, 2009.

AUTHORS BIOGRAPHY



Mrs.K. Anuradha completed her B.Sc (Applied Science) in 2000 and MCA in the year 2003. She is pursuing her Ph.D at Karpagam University, Coimbatore. She has published 7 papers in International Journals. She is currently working as Associate Professor in the Department of MCA, Karpagam College of Engineering, Coimbatore.

M. SAVARISELVI is currently pursing MCA in Karpagam College of engineering. Her area of interest is open source technologies, cloud computing.