ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 7, Issue 11, November 2018

Survey on Hadoop

Jambunathan.S[1], John Saferio.J[2], Harikaran.M[2]

Assistant Professor, Dept of Computer Science, Sri Krishna Arts and Science College, Coimbatore[1]

Student, III B.Sc Computer Science, Sri Krishna Arts and Science College, Coimbatore[2,3]

Abstract: Human life is depending on demand. This data is categories as "Big Data" due to its three Volume, Variety and Velocity. 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. Due to its specific nature of Big Data, it is stored in distributed file system architectures. Hadoop and HDFS by Apache is widely used for storing and managing Big 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 grouping of Big Data. These techniques are being adopted to be used in Map Reduce. In this paper we suggest basic over Hadoop and Hadoop Distributed File System (HDFS).

Keywords: Hadoop Distributed File System (HDFS), Relational Databases, Non-structured or semi-structured data model (NoSQL)

I. INTRODUCTION

1. BIG DATA

Big Data is a collection of large datasets that cannot be processed using traditional computing techniques. It is not a single technique or a tool, rather it involves many areas of business and technology. There are three dimensions to big data known as Volume, Variety and Velocity.

A)Volume: The quantity of generated and stored data. The size of the data determines the value and potential insight- and whether it can actually be considered big data or not. Collecting and analyzing more data helps make more educated business decisions. We want to store all data that have or might have business value. Don't throw anything away as you never know when a piece of data will be valuable for your organization. Flexibility in the ability to store data is also extremely important. Organizations require solutions that can scale easily. You might have only 1 Terabyte of data today but that may increase to 10 Terabytes in a few years and your data architecture must seamlessly and easily support scalability without "throw away" architectures.

B)Velocity: In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development. The rate of data collected, data which is flowing into our applications and systems, is increasing dramatically. Thousands, tens of thousands or even hundreds of thousands of critical business events are generated by our applications and systems every second. These business events are meaningful to us and have to be stored, cataloged and analyzed. Rapid data ingestion isn't the only challenge, users are demanding real time access to analytics based on up to date data. No longer can we provide users with reports based on yesterday's data. No longer can we rely on period nightly ETL jobs. Data needs to be fresh and immediately available to users for analytics as its being generated.

C)Variety: The type and nature of the data. This helps people who analyze it to effectively use the resulting insight. Traditional data sets used to be strictly structured. Either natively or after an ETL created structure – ETLs which are slow, non-scalable, difficult to change and prone to errors and failures. Nowadays, applications require to store different types of data, some structured yet some unstructured. Data generated from social networks, sensors, application logs, user interactions, geo-spatial data, etc. This is data which is much more complex and has to be made accessible for processing and analysis alongside more traditional data models. In addition, different applications with different data structures and use cases can benefit from different processing frameworks / paradigms. Some datasets require batch processing (such as recommendation engines) while other datasets rely on real time analytics (such as fraud detection). Flexibility in data access APIs – a "best of breed" approach can benefit users by making complex data easily accessible for everyone in our organization.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 7, Issue 11, November 2018

II. WHAT IS NOSQL DATABASE?

The solution to the challenges we described? The next-generation of NoSQL databases. Databases which try to address the "Volume, Velocity, and Variety" challenges by thinking outside the box.
Remember, relational databases are optimized for storing structured data, are difficult to scale and rely on SQL for data retrieval. They are optimized for some use case, but not all.
NoSQL databases, on the other hand, are designed to store and process large amount of data (Velocity,Volume) that is scalable (Volume), complex (Variety) and provide immediate access (Velocity) to fresh data.

A) Relational databases: Structured: data is stored in tables. Tables have data types, primary keys and constraints.
Transactional: data can be inserted and manipulated in "grouped units" = transactions. We can commit and rollback.
Versatile but limited: traditional relational databases can do OLTP, OLAP, DWH, Batch but are generally not specialized. Examples: Oracle, SQL Server, DB2, MySQL, PostgreSQL.
Do not easily scale-out: traditional relational database usually rely on a single database instance, scale-out requires manual shading, complex application level Data Access Layers (DALs) or expensive and specialized hardware. Well-known and easy to work with: everyone knows the RDBMS and SQL.

B) NoSQL databases: Non-structured or semi-structured data model: NoSQL databases usually provide a flexible data, supports un/semi-structured data, schema-less and support rapid data model changes. Some NoSQL databases provide native JSON support others provide a BigTable type data model.
Extremely Scalable: designed to be scalable from the ground up. Usually deployed in a cluster architecture to achieve easy and rapid scalability. Usually specialized: Specific NoSQL database technologies are designed for specific use cases. Examples: Hadoop, HBase, MongoDB, CouchBase, Cassandra, etc.
Varity of data retrieval and development APIs: each NoSQL database has its own individual query API and query language. Some even support SQLs, some do not.

III. HADOOP

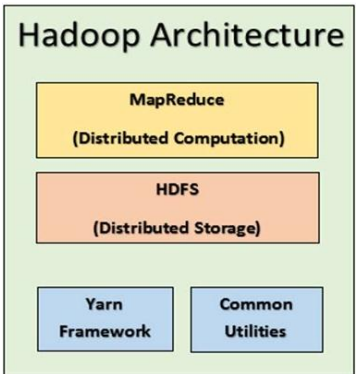
Hadoop runs the applications using the MapReduce algorithm, where the data is processed in parallel with others. It is used to develop the applications that could perform complete statistical analysis on large amounts of data.

3.1. Introduction to HADOOP

Hadoop is an open source framework written in programing language like java that allows distributed processing of large datasets across groups of computers using simple programming models. The Hadoop framework application works in an environment that provides individual storage and computation across groups of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

3.2. Hadoop Architecture:

At its core, Hadoop has two major layers namely:
(a) Processing/Computation layer (MapReduce), and
(b) Storage layer (Hadoop Distributed File System)



A)MapReduce: MapReduce is a programming model for writing distributed applications devised at websites for efficient processing of large amounts of data, on large groups of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 7, Issue 11, November 2018

How Does Hadoop Work?

Hadoop runs code across a group of computers. It includes the following tasks that Hadoop performs:

- Data is first divided into directories and files.
- Files are divided into same sized blocks of 128M and 64M .
- The files are then separated across various group nodes for further processing.
- Blocks are duplicative for handling hardware failure.
- After checking that the code was executed successfully.
- It performs the sort, that takes place between the map and reduce stages.
- Sending the sorted data to a certain computer.
- Writing the debugging logs for each job.

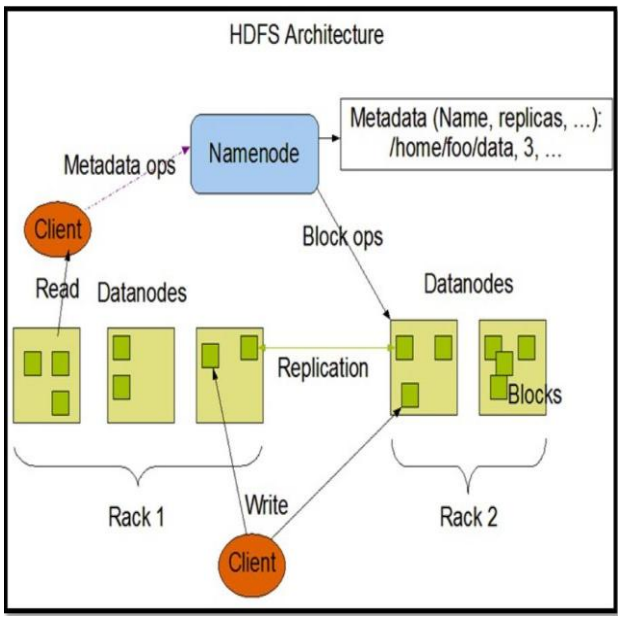
IV. HDFS

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault tolerant and designed using low-cost hardware. HDFS carry very large amount of data and provides easier access. To store such large data, the files are stored across different machines. These files are stored in necessary manner to rescue the system from possible data losses in case of any failure.

4.1. Features of HDFS

metadata including checksums for the block data and the block's generation stamp. The size of the data file equals the actual length of the block and does not require extra space to round it up to the nominal block size as in traditional file systems. The purpose of the data node isto verify the namespace ID and the software version of the data node. If either does not match that of the Name It is suitable for the separated storage and processing. Hadoop provides a command interface to interact with HDFS. Node the Data Node automatically shuts down. The namespace ID is assigned to the file system instance when it is formatted. The namespace ID is persistently. The built-in servers of name node and data node help users to easily check the status of group. Streaming access to file system data. HDFS provides file permission and authentication. stored on all nodes of the group. Nodes with a different namespace ID will not be able to join the group, thus preserving the integrity of the file system. The consistency of software of versions is important because

4.2. HDFS Architecture



A) Name Node: The HDFS is a hierarchy of files and directories. Files and directories are represented on the Name Node by inodes, which record attributes like modification, permission and access times, namespace and disk space. The file content is split into large blocks typically 128 megabytes, but user can select file-by-file and each block of the file is independently duplicated at multiple Data Nodes typically three, but user cans select file-by-file. The Name Node maintains the namespace tree and the mapping of file blocks to Data Nodes the physical

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 7, Issue 11, November 2018

B)Data Nodes: Each block duplicate on a Data Node is represented by two files in the local host's native file system. The first consistency of software versions is important because incompatible version may cause data corruption or loss, and on large groups of thousands of machines it is easy to overlook nodes that did not shut down properly prior to the software upgrade or were not available during the upgrade. A Data Node that is newly initialized and without any ID is permitted to join the group and receive the group's namespace ID. After the handshake the Data Node registers with the Name Node. Data Nodes persistently store their individual storage IDs. The storage ID is an internal identifier of the Data Node, which makes it recognizable even if it is restarted with a different IP address or port. The storage ID is assigned to the Data Node when it registers with the Name Node for the first time and never changes after that. A Data Node identifies block duplicates in its possession to the Name Node by sending a block report. A block report contains the block id, the generation stamp and the length for each block duplicate the server hosts. The first block report is sent immediately after the Data Node registration. Subsequent block reports are sent every hour and provide the Name Node with an up-to date view of where block duplicates are located on the group. During normal operation Data Nodes send heartbeats to the Name Node to confirm that the Data Node is operating and the block duplicates it hosts are available. The default heartbeat interval is three seconds. If the Name Node does not receive a heartbeat from a Data Node in ten minutes the Name Node considers the Data Node to be out of service and the block duplicates hosted by that Data Node to be unavailable. The Name Node then schedules creation of new duplicates of those blocks on other Data Nodes. Heartbeats from a Data Node also carry information about total storage capacity, fraction of storage in use, and the number of data transfers currently in progress. These statistics are used for the Name Node's space allocation and load balancing decisions. The Name Node does not directly call Data Nodes. It uses replies to heartbeats to send instructions to the Data Nodes. The instructions include commands to:

- Duplicate blocks to other nodes;
- Remove local block duplicates;
- Re-register or to shut down the node;
- Send an immediate block report.

These commands are important for maintaining the overall system integrity and therefore it is critical to keep heartbeats frequent even on big groups. The Name Node can process thousands of heartbeats per second without affecting other Name Node operations.

C) Check point Node: The Name Node in HDFS, has a primary role serving client requests, can alternatively execute either of two roles, Checkpoint Node or a Backup Node. The Checkpoint Node combines the existing checkpoint and journal to create a new checkpoint and an empty journal. The Checkpoint Node usually runs on a different host from the Name Node since it has the same memory requirements as the Name Node. It downloads the current checkpoint and journal files from the Name Node, and merges them locally, and returns the new checkpoint back to the Name Node. Creating periodic checkpoints is one way to protect the file system metadata. The system can start from the most recent checkpoint if all other persistent copies of the namespace image or journal are unavailable. Creating a checkpoint lets the Name Node truncate the tail of the journal when the new checkpoint is uploaded to the Name Node. HDFS groups run for prolonged periods of time without restarts during which the journal constantly grows. If the journal grows very large, the probability of loss or corruption of the journal file increases. Also, a very large journal extends the time required to restart the Name Node.

D) Backup Node: A recently introduced feature of HDFS is the Backup Node. Like a Checkpoint Node, the Backup Node is capable of creating periodic checkpoints, but in addition it maintains a memory, up-to-date image of the file system namespace that is always synchronized with the state of the Name Node. The Backup Node accepts the journal stream of namespace transactions from the active Name Node, saves them to its own storage directories, and applies these transactions to its own namespace image in memory. The Name Node treats the Backup Node as a journal store the same as it treats journal files in its storage directories. If the Name Node fails, the Backup Node's image in memory and the checkpoint on disk is a record of the latest namespace state. The Backup Node can create a checkpoint without downloading checkpoint and journal files from the active Name Node, since it already has an up-to-date namespace image in its memory. This makes the checkpoint process on the Backup Node more efficient as it only needs to save the namespace into its local storage directories. The Backup Node can be viewed as a read-only Name Node. It contains all file system metadata information except for block locations. It can perform all operations of the regular Name Node that do not involve modification of the namespace or knowledge of block locations. Use of a Backup Node provides the option of running the Name Node without persistent storage, delegating responsibility for the namespace state persisting to the Backup Node.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 7, Issue 11, November 2018

REFERENCES

[1]. Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler Yahoo! Sunnyvale, California USA {Shv, Hairong, SRadia, Chansler } @Yahoo-Inc.com
[2]. Apache Hadoop google .com
[3]. David Yahalom, CTO NAYA Technologies davidy@naya-tech.co.il www.naya-tech.com
[4]. Hadoop @ Tutorialsoint.com
[5]. 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.
[6]. Hadoop Sorts a Petabyte in 16.25 Hours and a Terabyte in 62 Seconds. May 2009. http://developer.yahoo.net/blogs/hadoop/2009/05/hadoop_sorts_a_petab yte_in_162.html
[7]. 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.
[8]. 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.
[9]. 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.
[10]. K. V. Shvachko, "HDFS Scalability: The limits to growth," ;login:. April 2010
[11]. Hadoop : The Definitive Guide Author : Tom White
[12]. Hadoop In Practice Author : Alex Holmes
[13]. Hadoop In Action Author : Chuck Lam
[14]. Hadoop Operations Author : Eric Sammers
[15]. Pro Hadoop Author : Jason Venner
[16]. Hadoop Beginner's Guide Author : Garry Turkington
[17]. Optimizing Hadoop For MapReduce Author : Khaled Tannir
[18]. Scaling Of Big Data With Hadoop And Solr Author : Hrishikesh
[19]. Hadoop Operations and Cluster Management Cookbook Author : Shumin Guo
[20]. Hadoop Real World Solution Cookbook Author : Jonathan Owens, Brian Femiano, Jon Lentz