ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Study of Apache Hadoop

Uma Patel, Rakesh Patel, Nimita Patel*
*Student,B.E.(IT), Kirodimal Institute of Technology,Raigarh(C.G.),India
Lecturer,Department of Information Technology, Kirodimal Institute of Technology Raigarh(C.G.),India
Student,B.E.(IT), Kirodimal Institute of Technology,Raigarh(C.G.),India

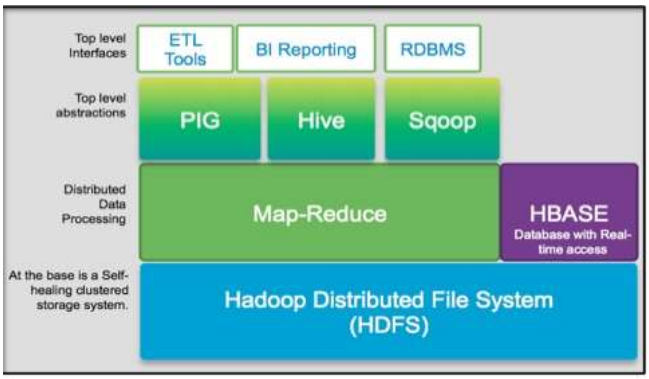
Abstract

Apache Hadoop is an open-source software framework for distributed storage and distributed processing of Big Data on clusters of commodity hardware. . The settings for the Hadoop environment are critical for deriving the full benefit from the rest of the hardware and software. The Distribution for Apache Hadoop* software includes Apache Hadoop* and other software components optimized to take advantage of hardware-enhanced performance and security capabilities.The Apache Hadoop project defines HDFS as "the primary storage system used by Hadoop applications" that enables reliable ,extremely rapid computations. Its Hadoop Distributed File System (HDFS) splits files into large blocks (default 64MB or 128MB) and distributes the blocks amongst the nodes in the cluster. Hadoop uses a distributed user-level filesystem. It takes care of storing data -- and it can handle very large amount of data.

Keywords: Apache hadoop.

Introduction

Apache Hadoop is an open source software from Apache Software Foundation. Apache Hadoop, and Hadoop are trademarks of The Apache Software Foundation. Used with permission. No endorsement by The Apache Software Foundation is implied by the use of these marks we implemented a low-cost, fully realized big data platform based on the Intel® Distribution for Apache Hadoop* software. It is an open source software stack that runs on a cluster of machines. Hadoop provides distributed storage and distributed processing for very large data sets. It is an Apache project released under Apache Open Source License v2.0.This license is very commercial friendly. Originally Hadoop was developed and open sourced by Yahoo. Now Hadoop is developed as an Apache Software Foundation project and has numerous contributors from Cloudera, Horton Works, Facebook, etc. Hadoop is open source. The software is free. Hadoop runs on a cluster of machines. The cluster size can be anywhere from 10 nodes to 1000s of nodes. For a large cluster, the hardware costs will be significant. The cost of IT / OPS for standing up a large Hadoop cluster and supporting it will need to be factored in. Since Hadoop is a newer technology, finding people to work on this ecosystem is not easy.



Architecture of hadoop

History

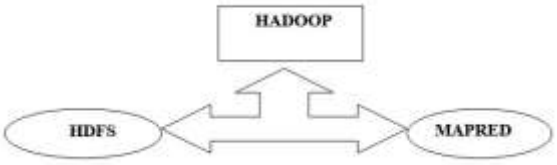
Hadoop was created by Doug Cutting and Mike Cafarella in 2005. Cutting, who was working at Yahoo! at the time, named it after his son's toy elephant. It was originally developed to support distribution for the Nutch search engine project.

Component of hadoop:

Hadoop provides two components:-

1. *Hadoop Distributed File System (HDFS)*
2. *MapReduce*.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114



Hadoop = HDFS + MapReduce
Hadoop provides two things : Storage & Compute.storage is provided by *Hadoop Distributed File System (HDFS)*. Compute is provided by *MapReduce*.

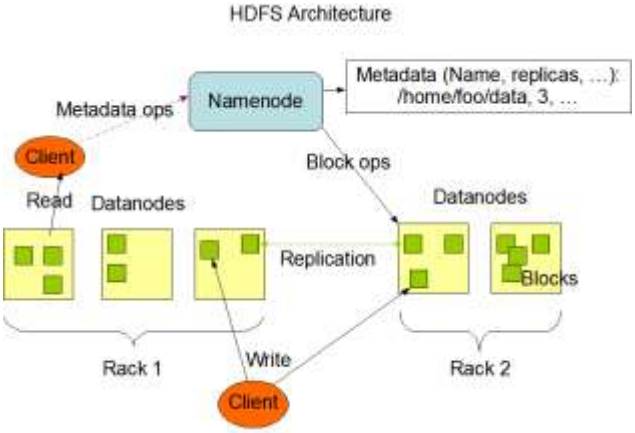
It consists of two parts: Hadoop Distributed File System (HDFS), which is modeled after Google's GFS, and Hadoop MapReduce, which is modeled after Google's MapReduce.

Hadoop distributed file system (HDFS)

HDFS is the 'file system' or 'storage layer' of Hadoop. It takes care of storing data and it can handle very large amount of data.The Hadoop Distributed File System (HDFS) is a distributed file system designed to run oncommodity hardware. It has many similarities with existing distributed file systems. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides highthrough put access to application data and is suitable for applications that have large data sets.

The Hadoop Distributed File System (HDFS) is one of many different components and projects contained within the community Hadoop™ ecosystem. The Apache Hadoop project defines.HDFS as: "the primary storage system used by Hadoop applications. HDFS creates multiple replicas of data blocks and distributes them on compute nodes throughout a cluster to enable reliable, extremely rapid computations.

HDFS Architecture



HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software. Usage of the highly portable Java language means that HDFS can be
deployed on a wide range of machines. A typical deployment has a dedicated machine that
runs only the NameNode software. Each of the other machines in the cluster runs one
instance of the DataNode software

NameNode and DataNode

The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case. HDFS does not support hard links or soft links. However, the HDFS architecture does not preclude implementing these features. HDFS is implemented by two services: the *NameNode* and *DataNode*.The NameNode maintains the file system namespace. The *NameNode* is responsible for maintaining the HDFS directory tree, and is a centralized service in the cluster operating on a single node. Any change to the file system namespace or its properties is recorded by the NameNode. An application can specify the number of replicas of a file that should be maintained by HDFS. The number of copies of a file is called the replication factor of that file. This information is stored by the NameNode. Clients contact the NameNode in order to perform common filesystem operations, such as open, close, rename, and delete. The NameNode does not store HDFS data itself, but rather maintains a mapping between HDFS file name, a list of blocks in the file, and the
DataNode(s) on which those blocks are stored.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

The file system namespace

HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories. The file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from one directory to another, or rename a file. HDFS does not yet implement user quotas or access permissions. HDFS does not support hard links or soft links. However, the HDFS architecture does not preclude implementing these features. The NameNode maintains the file system namespace. Any change to the file system namespace or its properties is recorded by the NameNode. An application can specify the number of replicas of a file that should be maintained by HDFS. The number of copies of a file is called the replication factor of that file. This information is stored by the NameNode.

Data replication

HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once and have strictly one writer at any time. The NameNode makes all decisions regarding replication of blocks. It periodically receives a Heartbeat and a Blockreport from each of the DataNodes in the cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly. A Blockreport contains a list of all blocks on a DataNode.

The Benefits of HDFS-

There is little debate that HDFS provides a number of benefits for those who choose to use it.Below are some of the most commonly cited.

Built-In Redundancy and Failover-

HDFS supplies out-of-the-box redundancy and failover capabilities that require little to no manual intervention (depending on the use case). Having such features built into the storage layer allows system administrators and developers to concentrate on other responsibilities versus having to create monitoring systems and/or programming routines to compensate for another set of storage software that lacks those capabilities. Moreover, with downtime being a real threat to many modern businesses' bottom line, features that minimize outages and contribute to keeping a batch analytic data store up, operational, and feeding any online system that requires its input are welcomed by both IT and business professionals.

Big Data Capable-

The hallmark of HDFS is its ability to tackle big data use cases and most of the characteristics that comprise them (data velocity, variety, and volume). The rate at which HDFS can supply data to the programming layers of Hadoop equates to faster batch processing times and quicker answers to complex analytic questions.

Portability-

Any tenured data professional can relay horror stories of having to transfer, migrate, and convert huge data volumes between disparate storage/software vendors. One benefit of HDFS is its portability between various Hadoop distributions, which helps minimize vendor lock-in.

Cost-Effective-

As previously stated, HDFS is open source software, which translates into real cost savings for its users. As many companies can attest, high-priced storage solutions can take a significant bite out of IT budgets and are many times completely out of reach for small or startup companies .Other benefits of HDFS exist, but the four above are the primary reasons why many users deploy HDFS as their analytic storage solution.

Map Reduce-

Map Reduce takes care of distributed computing. It reads the data, usually from its storage,A Hadoop MapReduce job mainly consists of two user-defined functions: map and reduce. The input of a Hadoop MapReduce job is a set of key-value pairs (k, v) and the map function is called for each of these pairs. The map function produces zero or more intermediate key-value pairs (k′, v′). Then, the Hadoop MapReduce framework groups these intermediate key-value pairs by intermediate key k′ and calls the reduce function for each group. Finally, the reduce function produces zero or more aggregated results. The beauty of Hadoop MapReduce is that users usually only have to define the map and reduce functions. The framework takes care of everything else such as parallelisation and failover. The Hadoop MapReduce framework utilizes a distributed file system to read and write its data. Typically, Hadoop MapReduce uses the Hadoop Distributed File System

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

(HDFS), which is the open source counterpart of the Google File System . Therefore, the I/O performance of a Hadoop MapReduce job strongly depends on HDFS.In the MapReduce model, computation is divided into a *map* function and a *reduce* function. The map function takes a key/value pair and produces one or more intermediate key/value pairs. The reduce function then takes these intermediate key/value pairs and merges all values corresponding to a single key. The map function can run independently on each key/value pair, exposing enormous amounts of parallelism. Similarly, the reduce function can run independently on each intermediate key, also exposing significant parallelism.

MapReduce has Mappers and Reducers
MapReduce splits computation into multiple tasks. They are called Mappers and Reducers

Mapper
The 'sorter' (the girl asking 'how old are you') only concerned about sorting people into appropriate groups (in our case, age). She isn't concerned about the next step of compute.
In MapReduce parlance the girl is known as **MAPPER.**

Reducer:-
Once the participants are sorted into appropriate age groups, then the guy wearing 'bowtie' just interviews that particular age group to produce the final result for that group .There are few subtle things happening here:

- The result for one age group is not influenced by the result of other age group. So they can be processed in parallel.
- we can be certain that each group has all participants for that group. For example, all 20 something's are in the group 20s. If the mapper did her job right, this would be the case.
- With these assumptions, the guy in bowtie can produce a result for a particular age group, independently

The benefits of MapReduce programming:-
So what are the benefits of MapReduce programming? As you can see, it summarizes a lot of the experiences of scientists and practitioners in the design of distributed processing systems. It resolves or avoids several complications of distributed computing. It allows unlimited computations on an unlimited amount of data. It actually simplifies the developer's life. And, although it looks deceptively simple, it is very powerful, with a great number of sophisticated (and profitable) applications written in this framework. In the other sections of this book we will introduce you to the practical aspects of MapReduce implementation. We will also show you how to avoid it, by using higher-level tools, 'cause not everybody likes to write Java code. Then you will be able to see whether or not Hadoop is for you, or even invent a new framework. Keep in mind though that other developers are also busy inventing new frameworks, so hurry to read more.

Hadoop Customization
A user can customize and optimize a Hadoop MapReduce job by supplying additional functions besides just map and reduce. In this section, we consider whether users are exploiting this feature in practice.

Job Customization
Most job customizations are related to ways of partitioning data and aggregating data from earlier stages:
Combiner: The Combiner performs partial aggregation during the local sort phase in a map task and a reduce task. In general, if the application semantics support it, a combiner is recommended. In OPENCLOUD, 62% of users have used this optimization at least once. In M45 and WEB MINING clusters, 43% and 80% of users have used it respectively.
Secondary Sort: This function is applied during the reduce phase. By grouping different reduce keys for a single reduce call, secondary sort allows users to implement an optimized join algorithm as well as other complex operations. In the M45 cluster, no user applied a secondary sort. In the WEB MINING cluster, only one user used secondary sort, and that was through the use of Pig, which implements certain join algorithms using secondary sort. In OPENCLOUD, 14% of users have used secondary sort, perhaps suggesting a higher level of sophistication or more stringent performance requirements.
Custom Partitioner: A user can also have full control over how to redistribute map output to the reduce tasks using a custom partitioner. In the OPENCLOUD cluster, as many as 35% of users have used a custom partitioner. However, only two users in the M45 cluster and one user in the WEB MINING cluster applied a custom partitioner.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Custom Input and Output Format: Hadoop provides an InputFormat and OutputFormat framework to simplify handling of custom data formats and non-native storage systems. In OPENCLOUD, 27% of users applied a custom input format at least once and 10% of users applied a custom output format. In M45, only 4 users applied a custom input format and only 1 user applied a custom output format. In WEB MINING, only one user applied a custom input format and none applied a custom output format. In general, job customizations help with performance and thus a visible fraction of users leverage them, especially the optional combiners. OPENCLOUD users tend to use more optimization techniques than users of the other two clusters. Configuration Tuning Hadoop exposes a variety of configuration parameters for tuning performance and reliability. Here we discuss a few configuration parameters that are typically considered important for performance and faulttolerance .

Failure Parameters: Users can control how failures are handled as well as erroneous inputs. In OPENCLOUD, 7 users explicitly specified a higher threshold to retry failed tasks, 6 users specified a higher "skip" to ignore bad input records, and 1 user specified a higher threshold in the number of tolerable failed tasks. In M45, 3 users set a higher threshold in the number of tolerable failed tasks. All WEB MINING users stayed with cluster default values.

Java Virtual Machine (JVM) Option: The native Hadoop MapReduce interface is implemented in Java. If a map or reduce task requires a large memory footprint, the programmer must manually adjust the heap and stack sizes: 29 OPENCLOUD users, 11 M45 users and 3 WEB MINING cluster users have changed default JVM option for their jobs. Speculative Execution: Speculative execution is the default mechanism to handle straggler tasks. Only two users from OPENCLOUD and M45 have changed the cluster default value for their applications. We discuss speculative execution in detail in Section 6.3.

Sort Parameters: Hadoop runs a merge sort at the end of the map phase and just before the reduce phase. There are four parameters that directly related to those sorts. Two users of WEB MINING cluster have adjusted io.sort.mb parameter to 200. Only one user of M45 cluster have adjusted io.sort.mb to 10. Other than that, all users used the cluster default values.

HDFS Parameters: The HDFS block size and replication factor affect the behavior of writing the final output of a MapReduce job. In OPENCLOUD, 11 users have tried different values for replication factor. In M45, two users have adjusted block size and only one user tried a different replication factor. Other than these, all users kept the cluster default values. In summary, users tend to tune parameters directly related to failures. JVM options are used to prevent "Out of Memory" errors. By talking with administrators of OPENCLOUD, we learned that many of their users explicitly tuned these options in response to poor failure behaviors. In contrast, users rarely tune parameters related to performance, perhaps because their performance requirements were generally being met, or perhaps because these parameters are more difficult to understand and manipulate.

Job optimization

One of the major advantages of Hadoop MapReduce is that it allows non-expert users to easily run analytical tasks over big data. Hadoop MapReduce gives users full control on how input data sets are processed. Users code their queries using Java rather than SQL. This makes Hadoop MapReduce easy to use for a larger number of developers: no background in databases is required; only a basic knowledge in Java is required. However, Hadoop MapReduce jobs are far behind parallel databases in their query processing efficiency. Hadoop MapReduce jobs achieve decent performance through scaling out to very large computing clusters. However, this results in high costs in terms of hardware and power consumption.Therefore, researchers have carried out many research works to effectively adapt the query processing techniques found in parallel databases to the context of Hadoop MapReduce.

Data layouts and indexes

One of the main performance problems with Hadoop MapReduce is its physical data organization including data layouts and indexes.

Data layouts: Hadoop MapReduce jobs often suffer from a roworiented layout. The disadvantages of row layouts have been thoroughly researched in the context of column stores . However, in a distributed system, a pure column store has severe drawbacks as the data for different columns may reside on different nodes leading to high network costs. Thus, whenever a query references more than one attribute, columns have to be sent through the network in order to merge different attributes values into a row (tuple re-construction). This can significantly decrease the performance of Hadoop MapReduce jobs. Therefore, other, more effective data layouts have been proposed in the literature for Hadoop MapReduce

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Conclusion

We find that in the three workloads, a majority of users submitted many small single stage applications implemented in Java, although the rest of workloads are highly diverse in application styles and data processing characteristics. We see underuse of Hadoop features, extensions and optimization tools. Our conclusion is that the use of Hadoop for academic research is still in its adolescence. Easing the use of Hadoop, and improving system designs subject to changing use cases are crucial research directions for future. Data confidentiality through encryption and decryption performed without a performance penalty in the storage layer Hadoop Distributed File System* (HDFS) taking full advantage of enhancements provided Advanced Encryption Standard New Instructions .The MapReduce programming model has been successfully used at Google for many different purposes. We attribute this success to several reasons.

References

1. HDFS Java API: http://hadoop.apache.org/core/docs/current/api/
2. HDFS source code: http://hadoop.apache.org/core/version_control.html
3. Apache hadoop.apache.org [http://hadoop.apache.org]
4. Kai Ren1, YongChul Kwon2, Magdalena Balazinska2, Bill Howe2 .Hadoop's Adolescence: A Comparative Workload Analysis from Three Research Clusters(Parallel Data Laboratory
5. Carnegie Mellon UniversityPittsburgh, PA 15213-3890).
6. Hadoop,http://hadoop.apache.org/mapreduce/.
7. A. Floratou et al. Column-Oriented Storage Techniques for MapReduce. PVLDB, 4(7):419–429, 2011.
8. "Hadoop HDFS," Hadoop.Apache.org: http://hadoop.apache.org/hdfs/.