Asian Journal of Convergence in Technology *Volume1, Issue 5*
Issn No.:2350-1146, I.F-2.71

Hadoop-Based Data Management Service for Cloud

Supriya Kumbhar
Department of CSE
Solapur University
Solapur, Maharstra
supriya.kumbhar92@yahoo.com

Ms.Sajjan R.S.
Department of CSE
Solapur,University
Solapur,University
line 4-e-mail address if desired

***Abstract*— In this paper we are going to discuss Hadoop-Based data management Service For Cloud. Data security issues and challenges in cloud computing, and also discuss environment used for data processing file management. In this paper we discuss architecture of hadoop also prominent users. Hadoop is an open-source software framework for storing and processing big data in a distributed fashion on large clusters of commodity hardware. Essentially, it accomplishes two tasks: massive storage and faster processing. Hadoop, a community driven Apache project has provided an efficient and cost effective platform for large scale computation using the map-reduce methodology, pioneered by Google. Hadoop capabilities are also extended in a quest for intelligent decision making regarding the choice of the fittest services for federation in a federated cloud scenario, in addition to legally behavior regarding the geographical location of data storage.**

***Keywords*— *Big Data, Cloud Computing, Distributed Data, Hadoop, Map Reduce, Hadoop Distributed File System, Distributed Nodes, Web Services*.**

Introduction

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware[1]. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets[1]. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project.

Apache Hadoop is an distributed processing of Big Data on clusters of commodity hardware and open-source software framework for distributed storage. Its Hadoop Distributed File System (HDFS) splits files into large blocks and distributes the blocks amongst the nodes in the cluster[2]. For processing the data, the Hadoop Map/Reduce ships code (specifically Jar files) to the nodes that have the required data, and the nodes then process the data in parallel. This approach takes advantage of data locality[2], in contrast to conventional HPC architecture which usually relies on a parallel file system (compute and data separated, but connected with high-speed networking)[3].

It supports the running of applications on large clusters of commodity hardware. The Hadoop framework transparently provides both data motion and reliability to applications. Hadoop is written in the Java programming language. It is a top-level Apache project being built and used by a global community of contributors. Hadoop enables applications to work with thousands of computation-independent computers and petabytes of data. It was derived from Google's MapReduce and Google File System (GFS) papers. It implements a computational paradigm named reduce, where the application is divided into many small fragments of work, each of which may be executed or reexecuted in the cluster on any node. It provides a distributed file system that stores data on the compute nodes, providing very high aggregate bandwidth across the cluster. Both reduce and the distributed file system is designed so that node failures are automatically handled by the framework. Haddop are loosely connected machines sharing no hardware or memory segments, which implementation is generally done with clusters. In any sophisticated system, there is a mixture of structured and unstructured data which is hard to fit into a single database. Hadoop is designed to run on a large number of machines that don't share any disks or memory. Hadoop spreads data over the organization's servers by breaking it into pieces. Hadoop consists of the Hadoop Common which provides access to the file systems supported by Hadoop. The Hadoop are Common packaging contains the necessary JAR files and scripts needed to start Hadoop. The package also provides documentation and source code. A small cluster Hadoop will include a single master and multiple worker nodes. The master node consists of a DataNode, JobTracker, NameNode, and TaskTracker. A worker or slave node acts as both a TaskTracker and DataNode, though it is possible to have data-only worker

nodes, and compute-only worker nodes; these are normally only used in non-standard applications. It requires JRE 1.6 or higher. The standard startup and shutdown scripts require ssh to be set up between nodes in the cluster[4].

In a large cluster, the HDFS is manage through a dedicate NameNode server to host the, a secondary NameNode and file system index that can generate snapshots of the name node's memory structures, thus preventing filesystem reducing loss of data and corruption[5].

In organizations, numbers of servers both host directly attached storage and execute user applications. The Hadoop cluster needs to be set according to the processing required.
The main workflow is as follows:

- Load data into the cluster
- Analyze the data (Map Reduce)
- Store results into the cluster (HDFS writes)
- Read results from the cluster (HDFS reads)

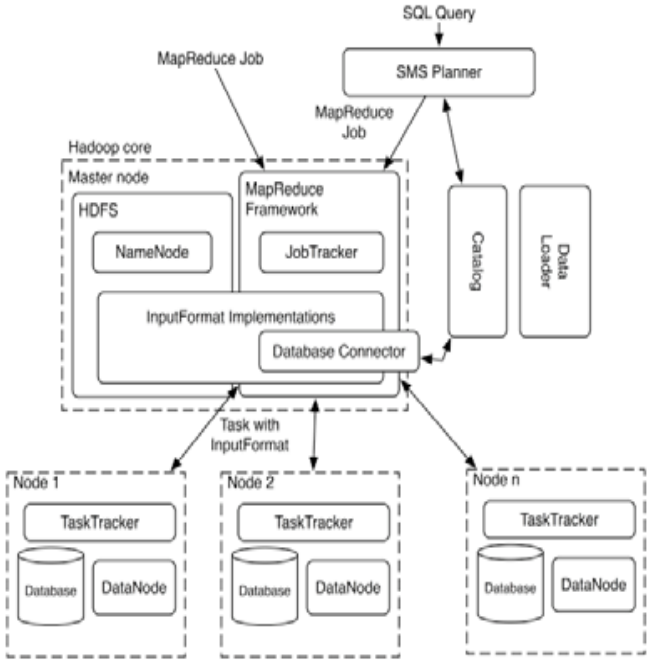

Fig 1 : *Structure of Hadoop*

The JobTracker must be assigned the instructions regarding the processing required in the form of Java code, which it then forwards to the corresponding nodes that contain the relevant data. Setting a Hadoop cluster requires the operator to strike a balance between performance and encryption, according to the necessity of either for data processing[5]. The emergence of a variety of Infrastructure as a Service provider has made it possible for companies or individuals to outsource their IT infrastructures with higher flexibility and dynamic reservation of resources according to their expected usage[4]. Another features offered by these providers is Storage as a Service, like the Amazon S3 offering. Storage systems and data management are still considered as very important challenges in the years to come, since current implementations show significant constraints for example with regard to federation capabilities and data lock in.

Furthermore, existing back-end solutions like Apache Hadoop have helped distributed infrastructures and forward the notion of parallel processing through an open source contribution with higher levels of performance, fault tolerance and ease of management. Their usage has been limited in internal infrastructures, for specific data intensive Processing tasks, mainly through the MapReduce framework.
The major aim of this to merge these fields through an innovative mechanism for a distributed and flexible Data Management Service (OPTIMIS Distributed FileSystem-ODFS), that will be able to offer Storage as a Service in a more transparent and flexible way than before. The ODFS is heavily based on Apache Hadoop and its underlying Distributed File System-HDFS and extends its functionalities from a back-end high performance parallel processing framework to a front-end, full-scale offering of Data as a Service. In order to do so, a number of functionalities have been implemented on top of Hadoop:

- Addition of RESTful interfaces in order to expose critical functionalities as services
- Extension of HDFS framework with regard to security, location management, data logging
- Ability to federate the Data Management Service to more than one IaaS providers, avoiding interoperability issues and data lock-in
- Ranking mechanisms for selection of services to federate from a data activity perspective
- Efficient management of data with regard to different scenarios, policies and infrastructure conditions.

I. HADOOP ARCHITECTURE

Hadoop consists of the *Hadoop Common* package, which provides filesystem and OS level abstractions, a MapReduce engine (either MapReduce/MR1 or YARN/MR2)[3]and the Hadoop Distributed File System (HDFS). The Hadoop Common package contains the necessary Java ARchive (JAR) files and scripts needed to start Hadoop. The package also provides source code, documentation, and a contribution section that includes projects from the Hadoop Community.

For effective scheduling of work, every Hadoop-compatible file system should provide location awareness: the name of the rack (more precisely, of the network switch) where a worker node is. Hadoop applications can use this information to run work on the node where the data is, and, failing that, on the same rack/switch, reducing backbone

traffic. HDFS uses this method when replicating data to try to keep different copies of the data on different racks. The goal is to reduce the impact of a rack power outage or switch failure, so that even if these events occur, the data may still be readable[4].

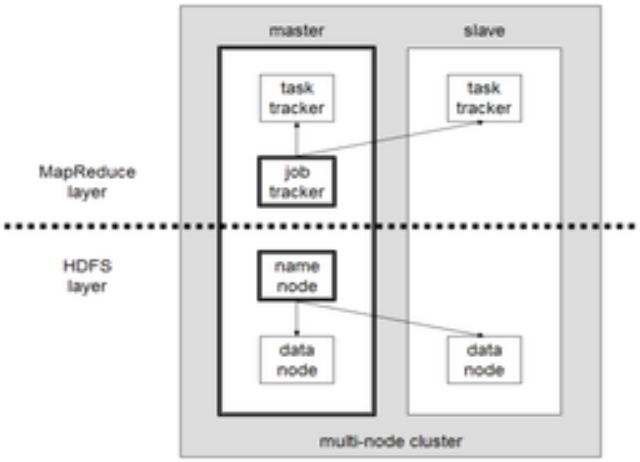

Fig 2 : *A multi-node Hadoop cluster*

A small Hadoop cluster includes a single master and multiple worker nodes. The master node consists of a JobTracker, TaskTracker, NameNode and DataNode. A slave or worker node acts as both a DataNode and TaskTracker, though it is possible to have data-only worker nodes and compute-only worker nodes. These are normally used only in nonstandard applications. Hadoop requires Java Runtime Environment (JRE) 1.6 or higher. The standard startup and shutdown scripts require that Secure Shell (ssh) be set up between nodes in the cluster[5].

In a larger cluster, the HDFS is managed through a dedicated NameNode server to host the file system index, and a secondary NameNode that can generate snapshots of the namenode's memory structures, thus preventing file-system corruption and reducing loss of data. Similarly, a standalone JobTracker server can manage job scheduling. In clusters where the Hadoop MapReduce engine is deployed against an alternate file system, the NameNode, secondary NameNode, and DataNode architecture of HDFS are replaced by the file-system-specific equivalents.

A. File System

a. Hadoop distributed file system

The **Hadoop distributed file system** (**HDFS**) is a distributed, scalable, and portable file-system written in Java for the Hadoop framework. A Hadoop cluster has nominally a single namenode plus a cluster of datanodes, although redundancy options are available for the namenode due to its criticality. Each datanode serves up blocks of data over the network using a block protocol specific to HDFS. The file system uses TCP/IP sockets for communication. Clients use remote procedure call (RPC) to communicate between each other.

HDFS stores large files (typically in the range of gigabytes to terabytes) across multiple machines. It achieves reliability by replicating the data across multiple hosts, and hence theoretically does not require RAID storage on hosts (but to increase I/O performance some RAID configurations are still useful). With the default replication value, 3, data is stored on three nodes: two on the same rack, and one on a different rack. Data nodes can talk to each other to rebalance data, to move copies around, and to keep the replication of data high. HDFS is not fully POSIX-compliant, because the requirements for a POSIX file-system differ from the target goals for a Hadoop application. The tradeoff of not having a fully POSIX-compliant file-system is increased performance for data throughput and support for non-POSIX operations such as Append[6].

HDFS added the high-availability capabilities, as announced for release 2.0 in May 2012, letting the main metadata server (the NameNode) fail over manually to a backup. The project has also started developing automatic fail-over.

The HDFS file system includes a so-called *secondary namenode*, a misleading name that some might incorrectly interpreted as a backup namenode for when the primary namenode goes offline. In fact, the secondary namenode regularly connects with the primary namenode and builds snapshots of the primary namenode's directory information, which the system then saves to local or remote directories. These checkpointed images can be used to restart a failed primary namenode without having to replay the entire journal of file-system actions, then to edit the log to create an up-to-date directory structure. Because the namenode is the single point for storage and management of metadata, it can become a bottleneck for supporting a huge number of files, especially a large number of small files. HDFS Federation, a new addition, aims to tackle this problem to a certain extent by allowing multiple namespaces served by separate namenodes.

An advantage of using HDFS is data awareness between the job tracker and task tracker. The job tracker schedules map or reduce jobs to task trackers with an awareness of the data location. For example: if node A contains data (x,y,z) and node B contains data (a,b,c), the job tracker schedules node B to perform map or reduce tasks on (a,b,c) and node A would be scheduled to perform map or reduce tasks on (x,y,z). This reduces the amount of traffic that goes over the network and prevents unnecessary data transfer. When Hadoop is used with other file systems, this advantage is not always available. This can have a significant impact on job-completion times, which has been demonstrated when running data-intensive jobs[7].

HDFS was designed for mostly immutable files and may not be suitable for systems requiring concurrent write-operations.

HDFS can be mounted directly with a Filesystem in Userspace (FUSE) virtual file system on Linux and some other Unix systems.

File access can be achieved through the native Java API, the Thrift API to generate a client in the language of the users' choosing (C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk, and OCaml), the command-line interface, browsed through the HDFS-UI webapp over HTTP, or via 3rd-party network client libraries[8].

b. Other file system

Hadoop works directly with any distributed file system that can be mounted by the underlying operating system simply by using a file:// URL; however, this comes at a price: the loss of locality. To reduce network traffic, Hadoop needs to know which servers are closest to the data; this is information that Hadoop-specific file system bridges can provide.

A number of third-party file system bridges have also been written, none of which are currently in Hadoop distributions. However, some commercial distributions of Hadoop ship with an alternative filesystem as the default, -specifically IBM and MapR.

- In 2009 IBM discussed running Hadoop over the IBM General Parallel File System.] The source code was published in October 2009[9].
- In April 2010, Parascale published the source code to run Hadoop against the Parascale file system.
- In April 2010, Appistry released a Hadoop file system driver for use with its own CloudIQ Storage product.
- In June 2010, HP discussed a location-aware IBRIX Fusion file system driver.
- In May 2011, MapR Technologies, Inc. announced the availability of an alternative file system for Hadoop, which replaced the HDFS file system with a full random-access read/write file system.

B. JobTracker and TaskTracker: the MapReduce engine

Above the file systems comes the MapReduce engine, which consists of one *JobTracker*, to which client applications submit MapReduce jobs. The JobTracker pushes work out to available *TaskTracker* nodes in the cluster, striving to keep the work as close to the data as possible. With a rack-aware file system, the JobTracker knows which node contains the data, and which other machines are nearby. If the work cannot be hosted on the actual node where the data resides, priority is given to nodes in the same rack. This reduces network traffic on the main backbone network. If a TaskTracker fails or times out, that part of the job is rescheduled. The TaskTracker on each node spawns off a separate Java Virtual Machine process to prevent the TaskTracker itself from failing if the running job crashes the JVM. A heartbeat is sent from the TaskTracker to the JobTracker every few minutes to check its status. The Job Tracker and TaskTracker status and information is exposed by Jetty and can be viewed from a web browser.

C. Other applications

The HDFS file system is not restricted to MapReduce jobs. It can be used for other applications, many of which are under development at Apache. The list includes the HBase database, the Apache Mahout machine learning system, and the Apache Hive Data Warehouse system. Hadoop can in theory be used for any sort of work that is batch-oriented rather than real-time, is very data-intensive, and benefits from parallel processing of data. It can also be used to complement a real-time system, such as lambda architecture.

II. HADOOP HOSTED IN THE CLOUD

Hadoop can be deployed in a traditional onsite datacenter as well as in the cloud. The cloud allows organizations to deploy Hadoop without hardware to acquire or specific setup expertise. Vendors who currently have an offer for the cloud include Microsoft, Amazon, and Google.

A. Hadoop on Microsoft Azure

Azure HDInsight is a service that deploys Hadoop on Microsoft Azure. HDInsight uses a Windows-based Hadoop distribution that was jointly developed with Hortonworks and allows programming extensions with .NET (in addition to Java). By deploying HDInsight in the cloud, organizations can spin up the number of nodes they want and only get charged for the compute and storage that is used.[10] Hortonworks implementations can also move data from the on-premises datacenter to the cloud for backup, development/test, and bursting scenarios.[10]

B. Hadoop on Amazon EC2/S3 services

It is possible to run Hadoop on Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3).[10] As an example The New York Times used 100 Amazon EC2 instances and a Hadoop application to process 4 TB of raw image TIFF data (stored in S3) into 11 million finished PDFs in the space of 24 hours at a computation cost of about $240 (not including bandwidth).

There is support for the S3 file system in Hadoop distributions, and the Hadoop team generates EC2 machine images after every release. From a pure performance perspective, Hadoop on S3/EC2 is inefficient, as the S3 file system is remote and delays returning from every write operation until the data is guaranteed not lost. This removes the locality advantages of Hadoop, which schedules work near data to save on network load.

C. Amazon Elastic MapReduce

Elastic MapReduce (EMR) was introduced by Amazon in April 2009. Provisioning of the Hadoop cluster, running and terminating jobs, and handling data transfer between EC2(VM) and S3(Object Storage) are automated by Elastic MapReduce. Apache Hive, which is built on top of Hadoop for providing data warehouse services, is also offered in Elastic MapReduce.

Support for using Spot Instances was later added in August 2011. Elastic MapReduce is fault tolerant for slave failures, and it is recommended to only run the Task Instance Group on spot instances to take advantage of the lower cost while maintaining availability.

Acknowledgment

It gives an immense pleasure to present a seminar report on topic "***HADOOP-BASED DATA MANAGEMENT SERVICE FOR CLOUD".***
I would like to acknowledge and extend gratitude to our Head of Department and seminar guide **Ms. Sajjan.R.S.** for her valuable guidance rendered in all phases by assistance, advice, stimulating suggestion and expert guidance towards working on this report.

References

[1] http://hadoop.apache.org/core/.

[2] "What is the Hadoop Distributed File System (HDFS)?". *ibm.com*. IBM. Retrieved 2014-10-30.

[3] Malak, Michael (2014-09-19). "Data Locality: HPC vs. Hadoop vs. Spark". *datascienceassn.org*. Data Science Association. Retrieved 2014-10-30.

[4] Tarush Jain et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 4 (3) , 2013, 530-532

[5] Intel Distribution for Apache Hadoop Tarush Jain, Rohan Somni *IT Department, Pune University, Pune, India.*

[6] Yaniv Pessach (2013). "Distributed Storage" (Distributed Storage: Concepts, Algorithms, and Implementations ed.). Amazon.com

[7] "Improving MapReduce performance through data placement in heterogeneous Hadoop Clusters" (PDF). Eng.auburn.ed. April 2010.

[8] "Mounting HDFS". Retrieved May 2014.

[9] "HADOOP-6330: Integrating IBM General Parallel File System implementation of Hadoop Filesystem interface". IBM. 2009-10-23.

[10] "HDInsight | Cloud Hadoop". Azure.microsoft.com. Retrieved 2014-07-22