Wuhan University of technology

School of computer science and technology

Distributed and Parallel Computing

FINAL REPORT:

Hadoop Ecosystem

13/07/2015

Submitted by: OUMOUSS EL MEHDI
Student No: 2014Y90100063
TEL No: 13125029535
Email: oumoussmehdi@hotmail.com
Submitted to: Prof. Dr. Yuan Jingling

Hadoop Ecosystem

Abstract

In the age of big data, dealing with large set of data is not any more possible using traditional computing techniques. Hadoop an open source framework enables distributed parallel processing of huge amounts of data across inexpensive, industry-standard servers that both store and process the data, and can scale without limits. In this report we are going to draw the map of the Hadoop Ecosystem in order to discover the different technologies that maintain and support Hadoop.

Keywords: Hadoop Ecosystem, Distributed Systems, MapReduce, HDFS, Big Data.

Introduction

A *distributed system* is a software system in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. Three significant characteristics of distributed systems are: concurrency of components, lack of a global clock, and independent failure of components. Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications [1].

A computer program that runs in a distributed system is called a distributed program.

1. What is Hadoop?

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. The Apache Hadoop software library created by Doug Cutting and Mike Cafarella in 2005, is a framework, mostly written in java that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

The core of Apache Hadoop consists of four main modules:

- Hadoop Common: contains libraries and utilities needed by other Hadoop modules.
- Hadoop Distributed File System (HDFS): A distributed file system that provides high-throughput access to application data (distributed storage).
- Hadoop YARN: a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications
- Hadoop MapReduce: A YARN-based system for parallel processing of large data sets (distributed computation).
-

Apache Hadoop's MapReduce and HDFS components were inspired by Google papers on their MapReduce and Google File System.

As a distributed program, 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. [2,3,4]

The building blocks of Hadoop

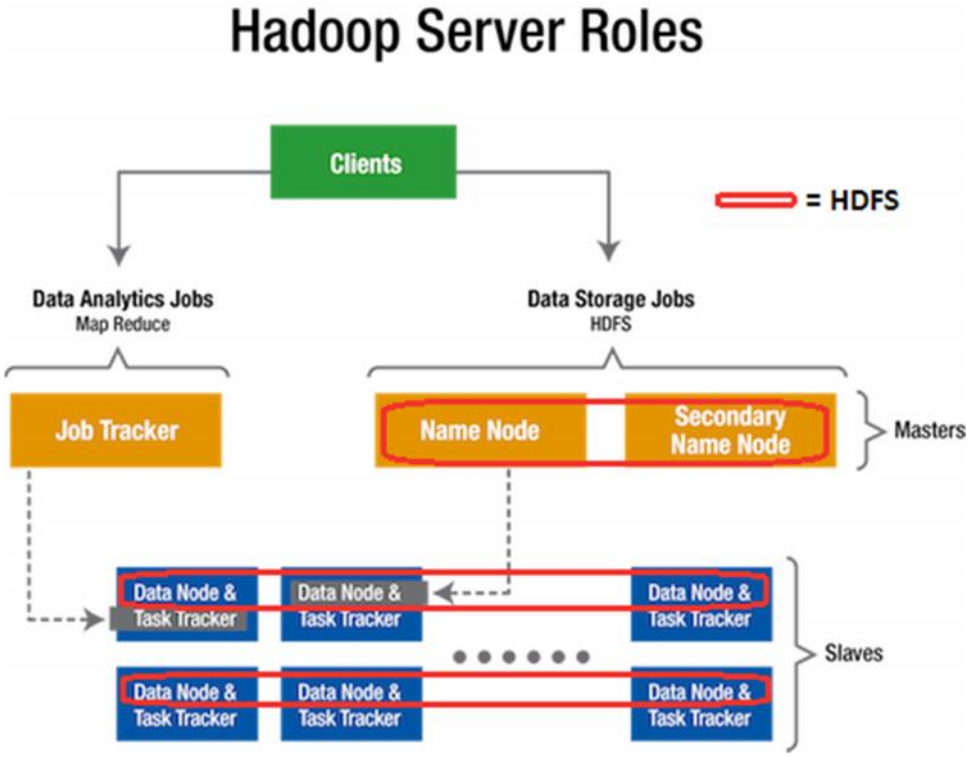

Figure1: Hadoop baisic Architecture [5]

We've pointed quickly the concepts of distributed storage and distributed computation above. Now let's see how Hadoop implements those ideas. On a fully configured cluster, "running Hadoop" means running a set of daemons, or resident programs, on the different servers in your network. These daemons have specific roles; some exist only on one server, some exist across multiple servers. The daemons include:

- NameNode
- DataNode
- Secondary NameNode
- JobTracker
- TaskTracker

NameNode: is the most vital of the Hadoop daemons. Hadoop employs a master/slave architecture for both distributed storage and distributed computation. The distributed storage system is called HDFS. The NameNode is the master of HDFS that directs the slave DataNode daemons to perform the low-level I/O tasks. The NameNode is the bookkeeper of HDFS; it keeps track of how your files are broken down into file blocks, which nodes store those blocks, and the overall health of the distributed filesystem.

The function of the NameNode is memory and I/O intensive. As such, the server hosting the NameNode typically doesn't store any user data or perform any computations for a MapReduce program to lower the workload on the machine. This means that the NameNode server doesn't double as a DataNode or a TaskTracker.

There is unfortunately a negative aspect to the importance of the NameNode—it's a single point of failure of your Hadoop cluster. For any of the other daemons, if their host nodes fail for software or hardware reasons, the Hadoop cluster will likely continue to function smoothly or you can quickly restart it. Not so for the NameNode.

DataNode: Each slave machine in your cluster will host a DataNode daemon to perform the grunt work of the distributed file system—reading and writing HDFS blocks to actual files on the local filesystem. When you want to read or write a HDFS file, the file is broken into blocks and the NameNode will tell your client which DataNode each block resides in. Your client communicates directly with the DataNode daemons to process the local files corresponding to the blocks. Furthermore, a DataNode may communicate with other DataNodes to replicate its data blocks for redundancy. This ensures that if any one DataNode crashes or becomes inaccessible over the network, you'll still be able to read the files. DataNodes are constantly reporting to the NameNode. Upon initialization, each of the DataNodes informs the NameNode of the blocks it's currently storing. After this mapping is complete, the DataNodes continually poll the NameNode to provide information regarding local changes as well as receive instructions to create, move, or delete blocks from the local disk.

Secondary NameNode: The Secondary NameNode (SNN) is an assistant daemon for monitoring the state of the cluster HDFS. Like the NameNode, each cluster has one SNN, and it typically resides on its own machine as well. No other DataNode or TaskTracker daemons run on the same server. The SNN differs from the NameNode in that this process doesn't receive or record any real-time changes to HDFS. Instead, it communicates with the NameNode to take snapshots of the HDFS metadata at intervals defined by the cluster configuration. As mentioned earlier, the NameNode is a single point of failure for a Hadoop cluster, and the SNN snapshots help minimize the downtime and loss of data. Nevertheless, a NameNode failure requires human intervention to reconfigure the cluster to use the SNN as the primary NameNode.

JobTracker: The JobTracker daemon is the liaison between your application and Hadoop. Once you submit your code to your cluster, the JobTracker determines the execution plan by determining which files to process, assigns nodes to different tasks, and monitors all tasks as they're running. Should a task fail, the JobTracker will automatically relaunch the task, possibly on a different node, up to a predefined limit of retries. There is only one JobTracker daemon per Hadoop cluster. It's typically run on a server as a master node of the cluster.

TaskTracker: As with the storage daemons, the computing daemons also follow a master/slave architecture: the JobTracker is the master overseeing the overall execution of a MapReduce job and the TaskTrackers manage the execution of individual tasks on each slave node. Each TaskTracker is responsible for executing the individual tasks that the JobTracker assigns. Although there is a single TaskTracker per slave node, each TaskTracker can spawn multiple JVMs to handle many map or reduce tasks in parallel. One responsibility of the TaskTracker is to constantly communicate with the JobTracker. If the JobTracker fails to receive a heartbeat from a TaskTracker within a specified amount of time, it will assume the TaskTracker has crashed and will resubmit the corresponding tasks to other nodes in the cluster.[6,7]

Note: The term "Hadoop" refers not just to the modules above, but also to the "ecosystem", or collection of additional software packages that can be installed on top of or alongside Hadoop.

2. Hadoop Ecosystem

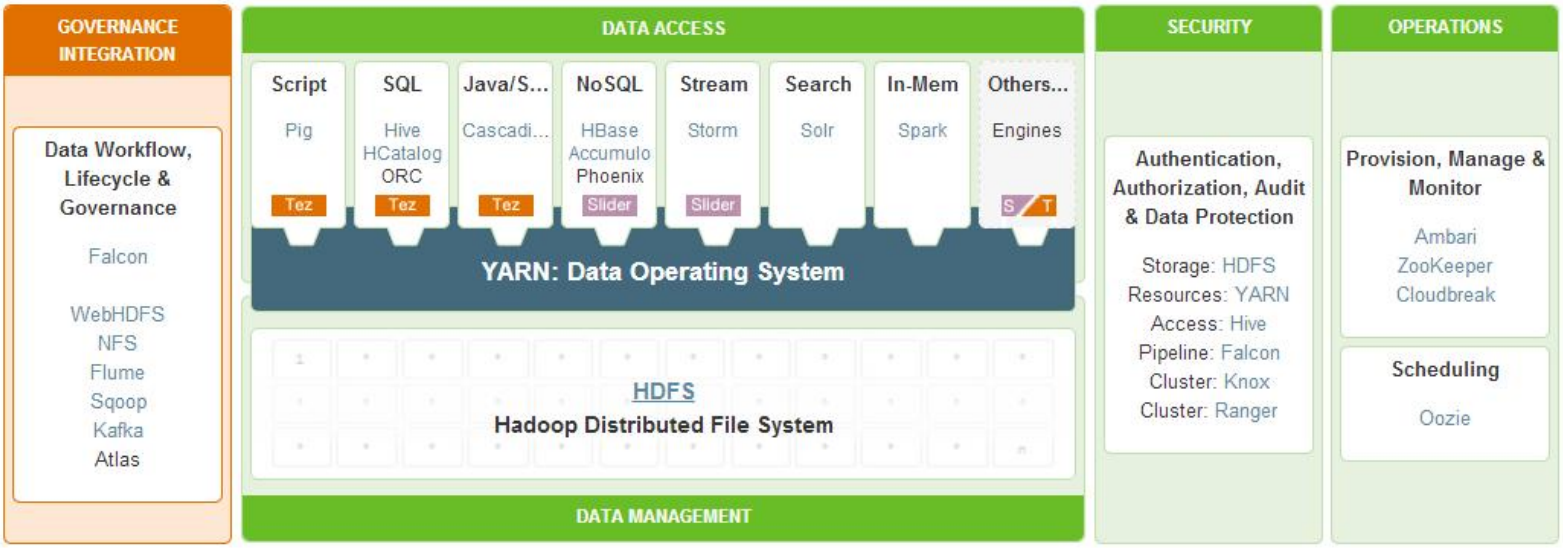

Figure 2: Hadoop Ecosystem [9]

2.1 Data Management

Hadoop Distributed File System (HDFS): A Java-based file system that provides scalable and reliable data storage that is designed to span large clusters of commodity servers. An HDFS cluster has two types of nodes operating in a master-worker pattern: a namenode (the master) and a number of datanodes (workers), which we presented explicitely above [8].

YARN and MapReduce 2

Apache Hadoop YARN is the pre-requisite for Enterprise Hadoop as it provides the resource management and pluggable architecture for enabling a wide variety of data access methods to operate on data stored in Hadoop with predictable performance and service levels.
YARN is a next-generation framework for Hadoop data processing extending MapReduce capabilities by supporting non-MapReduce workloads associated with other programming models. many of Hadoop engines (HBase, Accumuloe etc) can work across one set of data and resources thanks to YARN. YARN also provides flexibility for new and emerging data access methods, such as Apache Solr for search and programming frameworks such as Cascading.

MapReduce 2.0, or MR2, contains the same execution framework as MapReduce 1.0, but it is built on the scheduling/resource management framework of YARN. contrary to widespread misconceptions, YARN is not the same as MapReduce 2.0 (MRv2). Rather, YARN is a general framework which can support multiple instances of distributed processing applications, of which MapReduce 2.0 is one.

2.2 Data Access

Interact with your data in a wide variety of ways – from batch to real-time.

MapReduce:

MapReduce is a framework for writing applications that process large amounts of structured and unstructured data in parallel across a cluster of thousands of machines, in a reliable and fault-tolerant manner. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once you write an application in the MapReduce form, scalability is guaranteed.

How MapReduce Works?

A MapReduce job splits a large data set into independent chunks and organizes them into key, value pairs for parallel processing. This parallel processing improves the speed and reliability of the cluster, returning solutions more quickly and with greater reliability.

The Map function divides the input into ranges by the Input Format and creates a map task for each range in the input. The JobTracker distributes those tasks to the worker nodes. The output of each map task is partitioned into a group of key-value pairs for each reduce.

The Reduce function then collects the various results and combines them to answer the larger problem that the master node needs to solve. Each reduce pulls the relevant partition from the machines where the maps executed, then writes its output back into HDFS. Thus, the reduce is able to collect the data from all of the maps for the keys and combine them to solve the problem.

here is an example shown in the next figure, where we use the MapReduce paradigm to calculate word occurrence.

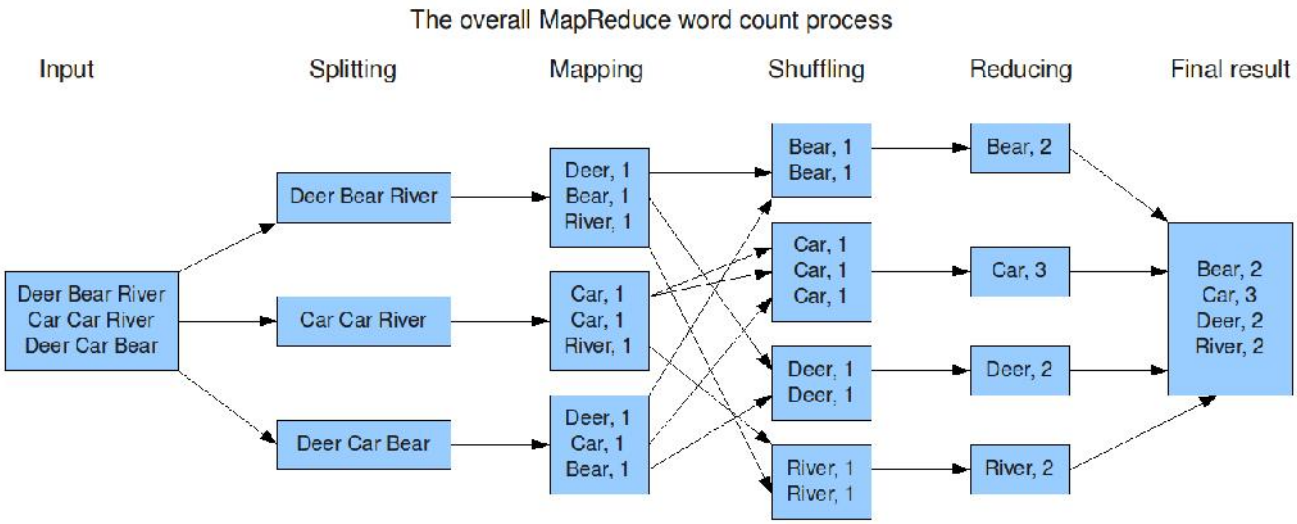

Figure 3: MapReduce process [10]

Apache Accumulo: Accumulo is a high performance data storage and retrieval system with cell-level access control. It is a scalable implementation of Google's Big Table design that works on top of Apache Hadoop, ZooKeeper, and Thrift. Apache Accumulo features a few novel improvements on the BigTable design in the form of cell-based access control and a server-side programming mechanism that can modify key/value pairs at various points in the data management process.

Apache HBase: A column-oriented NoSQL data storage system that provides random real-time read/write access to big data for user applications. Apache HBase is an open-source, distributed, versioned, non-relational database modeled after Google's. Just as Bigtable leverages the distributed data storage provided by the Google File System, Apache HBase provides Bigtable-like capabilities on top of Hadoop and HDFS.

Apache HCatalog: A table and metadata management service that provides a centralized way for data processing systems to understand the structure and location of the data stored within Apache Hadoop.

Apache Hive: Apache Hive is the most widely adopted data access technology, though there are many specialized engines. Built on the MapReduce framework, The Apache Hive data warehouse

software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL. At the same time this language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

Apache Kafka: Kafka is a fast and scalable publish-subscribe messaging system that is often used in place of traditional message brokers because of its higher throughput, replication, and fault tolerance.

Apache Mahout: Mahout provides scalable machine learning algorithms for Hadoop which aids with data science for clustering, classification and batch based collaborative filtering (recommendation).

Apache Pig: a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.at the present time, Pig's infrastructure layer consists of a compiler that produces sequences of Map-Reduce programs, for which large-scale parallel implementations already exist (e.g., the Hadoop subproject). Pig's language layer currently consists of a textual language called Pig Latin.

Apache Slider: A framework for deployment of long-running data access applications in Hadoop. Slider leverages YARN's resource management capabilities to deploy those applications, to manage their lifecycles and scale them up or down.

Apache Solr: Solr is the open source platform for searches of data stored in Hadoop. Solr enables powerful full-text search and near real-time indexing on many of the world's largest Internet sites.

Apache Spark: a free and open source distributed realtime computation system. Storm makes it easy to reliably process unbounded streams of data, doing for realtime processing what Hadoop did for batch processing. Storm is simple, can be used with any programming language. Storm has many use cases: realtime analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. Storm is fast: a benchmark clocked it at over a million tuples processed per second per node. It is scalable, fault-tolerant and guarantees your data will be processed.

Apache Storm: Storm is a distributed real-time computation system for processing fast, large streams of data adding reliable real-time data processing capabilities to Apache Hadoop 2.x

Apache Tez: Tez generalizes the MapReduce paradigm to a more powerful framework for executing a complex DAG (directed acyclic graph) of tasks for near real-time big data processing.

2.3 Data Governance & Integration

Quickly and easily load data, and manage according to policy.

Apache Falcon: Falcon is a data management framework for simplifying data lifecycle management and processing pipelines on Apache Hadoop®. It enables users to orchestrate data motion, pipeline processing, disaster recovery, and data retention workflows.

Apache Flume: Flume allows you to efficiently aggregate and move large amounts of log data from many different sources to Hadoop.

Apache Sqoop: Sqoop is a tool that speeds and eases movement of data in and out of Hadoop. It provides a reliable parallel load for various, popular enterprise data sources.

2.4 Security.

Address requirements of Authentication, Authorization, Accounting and Data Protection.

Apache Knox: The Knox Gateway provides a single point of authentication and access for Apache Hadoop services in a cluster. The goal of the project is to simplify Hadoop security for users who access the cluster data and execute jobs, and for operators who control access to the cluster.

Apache Ranger: Apache Ranger delivers a comprehensive approach to security for a Hadoop cluster. It provides central security policy administration across the core enterprise security requirements of authorization, accounting and data protection.

2.5 Operations.

Provision, manage, monitor and operate Hadoop clusters at scale.

Apache Ambari: A project is aimed at making Hadoop management simpler by developing software for provisioning, managing, and monitoring Apache Hadoop clusters. Ambari provides an intuitive, easy-to-use Hadoop management web UI backed by its RESTful APIs.

Apache Oozie: A workflow scheduler system to manage Apache Hadoop jobs. Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of actions. Oozie Coordinator jobs are recurrent Oozie Workflow jobs triggered by time (frequency) and data availabilty. Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs out of the box .

Apache ZooKeeper: A highly available system for coordinating distributed processes. Distributed applications use ZooKeeper to store and mediate updates to important configuration information [2,9].

3. Applications

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.

Hadoop has many commercial application, namely:

- Log and/or clickstream analysis of various kinds
- Marketing analytics
- Machine learning and/or sophisticated data mining
- Image processing
- Processing of XML messages
- Web crawling and/or text processing
- General archiving, including of relational/tabular data, e.g. for compliance

An other Hadoop application is the implementation of Hadoop in public cloud spaces such as IBM Bluemix. Microsoft Azure, Amazon Web Services (e.g. Amazon Elastic MapReduce) and Google Compute Engine.

4. Prominent users

Hadoop is used by all the big companies especially those who deals with bid data. for instance, The Yahoo! Search Webmap is a Hadoop application that runs on a Linux cluster with more than 10,000 cores and produced data that was used in every Yahoo! web search query. In 2010, Facebook claimed that they had the largest Hadoop cluster in the world with 21 PB of storage. Add to this that more than half of the Fortune 50 use Hadoop [3].

Conclusion

As we have seen, Hadoop with its different technologies represent a hole of ecosystem. Typically, thanks to the synchronization between the different components. The list of technologies we presented is not in any case an extensive one, because Hadoop Ecosystem knows fast evolution, in other word there is always new projects added to the Hadoop project.

References:

[1] Distributed computing, Wikipedia
https://en.wikipedia.org/wiki/Distributed_computing

[2] Welcome to Apache™ Hadoop®!
https://hadoop.apache.org/

[3] Apache Hadoop, Wikipedia
https://en.wikipedia.org/wiki/Apache_Hadoop

[4] The Hadoop Ecosystem, Cloudera
http://www.cloudera.com/content/cloudera/en/training/library/apache-hadoop-ecosystem.html

[5] Introduction to the Hadoop Software Ecosystem,
http://www.revelytix.com/?q=content/hadoop-ecosystem

[6] CHUCK LAM, Hadoop in Action, 2010, MANNING Publications Co.
www.wowebook.com

[7] Tom White, Hadoop: The Definitive Guide, Fourth Edition, Published by O'Reilly Media

[8] Clemens Neudecker, What is Hadoop? Hadoop Driven Digital Preservation, SCAPE

[9] Enterprise Hadoop: The Ecosystem of Projects
http://hortonworks.com/hadoop/

[10] XIAOCHONG ZHANG, A Simple Example to Demonstrate how does the MapReduce work,
http://xiaochongzhang.me/blog/?p=338