4.2 HADOOP

Hadoop is an open-source project of the Apache foundation. It is a framework written in Java, originally developed by Doug Cutting in 2005 who named it after his son's toy elephant. He was working with Yahoo then. It was created to support distribution for "Nutch", the text search engine. Hadoop uses Google's MapReduce and Google File System technologies as its foundation. Hadoop is now a core part of the computing infrastructure for companies such as Yahoo, Facebook, LinkedIn, Twitter, etc. Refer Figure 4.8.

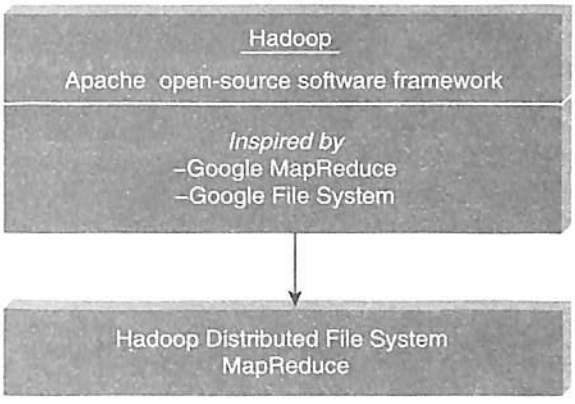


Figure 4.8 Hadoop.

4.2.1 Features of Hadoop

Let us cite a few features of Hadoop:

1. It is optimized to handle massive quantities of structured, semi-structured, and unstructured data, using commodity hardware, that is, relatively inexpensive computers.
2. Hadoop has a shared nothing architecture.
3. It replicates its data across multiple computers so that if one goes down, the data can still be processed from another machine that stores its replica.
4. Hadoop is for high throughput rather than low latency. It is a batch operation handling massive quantities of data; therefore the response time is not immediate.
5. It complements On-Line Transaction Processing (OLTP) and On-Line Analytical Processing (OLAP). However, it is not a replacement for a relational database management system.
6. It is NOT good when work cannot be parallelized or when there are dependencies within the data.
7. It is NOT good for processing small files. It works best with huge data files and datasets.

4.2.2 Key Advantages of Hadoop

Refer Figure 4.9 for a quick look at the key advantages of Hadoop. Some of them are as follows:

1. **Stores data in its native format:** Hadoop's data storage framework (HDFS – Hadoop Distributed File System) can store data in its native format. There is no structure that is imposed while keying in data or storing data. HDFS is pretty much schema-less. It is only later when the data needs to be processed that structure is imposed on the raw data.
2. **Scalable:** Hadoop can store and distribute very large datasets (involving thousands of terabytes of data) across hundreds of inexpensive servers that operate in parallel.

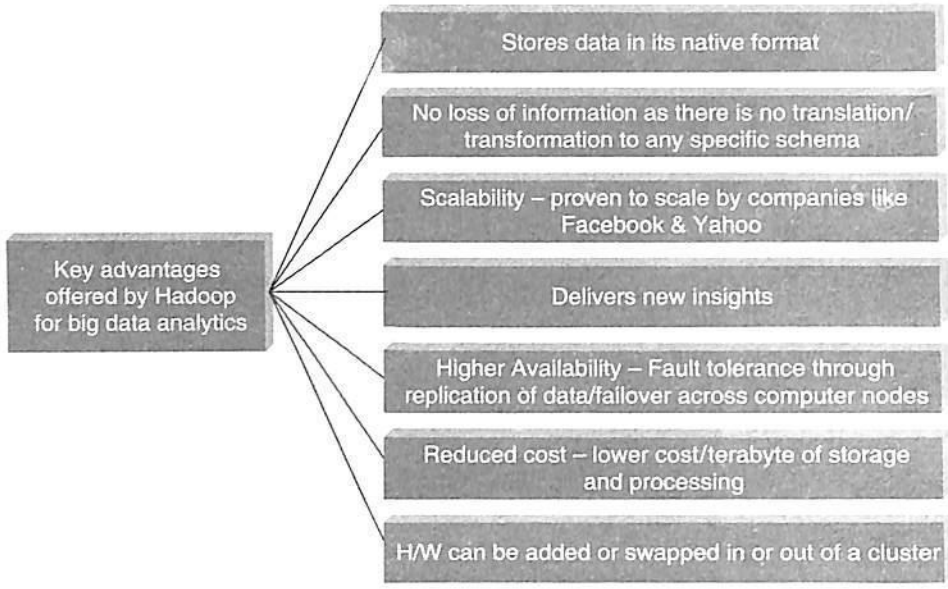


Figure 4.9 Key advantages of Hadoop.

3. **Cost-effective:** Owing to its scale-out architecture, Hadoop has a much reduced cost/terabyte of storage and processing.
4. **Resilient to failure:** Hadoop is fault-tolerant. It practices replication of data diligently which means whenever data is sent to any node, the same data also gets replicated to other nodes in the cluster, thereby ensuring that in the event of a node failure, there will always be another copy of data available for use.
5. **Flexibility:** One of the key advantages of Hadoop is its ability to work with all kinds of data: structured, semi-structured, and unstructured data. It can help derive meaningful business insights from email conversations, social media data, click-stream data, etc. It can be put to several purposes such as log analysis, data mining, recommendation systems, market campaign analysis, etc.
6. **Fast:** Processing is extremely fast in Hadoop as compared to other conventional systems owing to the "move code to data" paradigm.

   Hadoop has a shared-nothing architecture.

4.2.3 Versions of Hadoop

There are two versions of Hadoop available:

1. Hadoop 1.0
2. Hadoop 2.0

Let us take a look at the features of both. Refer Figure 4.10.

4.2.3.1 Hadoop 1.0

It has two main parts:

1. **Data storage framework:** It is a general-purpose file system called Hadoop Distributed File System (HDFS). HDFS is schema-less. It simply stores data files. These data files can be in just about any

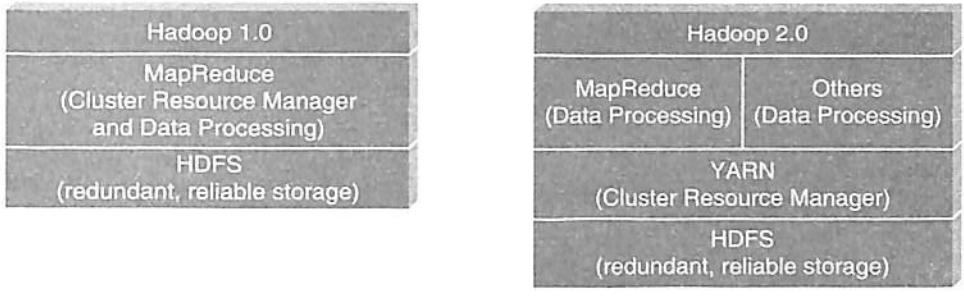


Figure 4.10 Versions of Hadoop.

format. The idea is to store files as close to their original form as possible. This is turn provides the business units and the organization the much needed flexibility and agility without being overly worried by what it can implement.

2. **Data processing framework:** This is a simple functional programming model initially popularized by Google as MapReduce. It essentially uses two functions: the MAP and the REDUCE functions to process data. The "Mappers" take in a set of key–value pairs and generate intermediate data (which is another list of key–value pairs). The "Reducers" then act on this input to produce the output data. The two functions seemingly work in isolation from one another, thus enabling the processing to be highly distributed in a highly-parallel, fault-tolerant, and scalable way.

There were, however, a few limitations of Hadoop 1.0. They are as follows:

1. The first limitation was the requirement for MapReduce programming expertise along with proficiency required in other programming languages, notably Java.
2. It supported only batch processing which although is suitable for tasks such as log analysis, large-scale data mining projects but pretty much unsuitable for other kinds of projects.
3. One major limitation was that Hadoop 1.0 was tightly computationally coupled with MapReduce, which meant that the established data management vendors were left with two options: Either rewrite their functionality in MapReduce so that it could be executed in Hadoop or extract the data from HDFS and process it outside of Hadoop. None of the options were viable as it led to process inefficiencies caused by the data being moved in and out of the Hadoop cluster.

Let us look at whether these limitations have been wholly or in parts resolved by Hadoop 2.0.

4.2.3.2 Hadoop 2.0

In Hadoop 2.0, HDFS continues to be the data storage framework. However, a new and separate resource management framework called **Y**et **A**nother **R**esource **N**egotiator (YARN) has been added. Any application capable of dividing itself into parallel tasks is supported by YARN. YARN coordinates the allocation of subtasks of the submitted application, thereby further enhancing the flexibility, scalability, and efficiency of the applications. It works by having an ApplicationMaster in place of the erstwhile JobTracker, running applications on resources governed by a new NodeManager (in place of the erstwhile TaskTracker). ApplicationMaster is able to run any application and not just MapReduce.

This, in other words, means that the MapReduce Programming expertise is no longer required. Furthermore, it not only supports batch processing but also real-time processing. MapReduce is no longer the only data processing option; other alternative data processing functions such as data standardization, master data management can now be performed natively in HDFS.

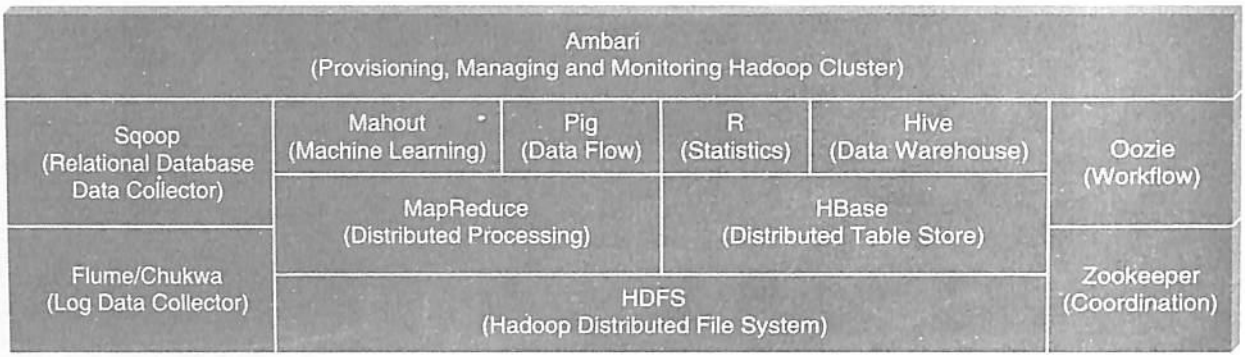


Figure 4.11 Hadoop ecosystem.

4.2.4 Overview of Hadoop Ecosystems

The components of the Hadoop ecosystem are shown in Figure 4.11.

There are components available in the Hadoop ecosystem for data ingestion, processing, and analysis.

Data Ingestion → Data Processing → Data Analysis

Components that help with Data Ingestion are:

1. Sqoop
2. Flume

Components that help with Data Processing are:

1. MapReduce
2. Spark

Components that help with Data Analysis are:

1. Pig
2. Hive
3. Impala

HDFS

It is the distributed storage unit of Hadoop. It provides streaming access to file system data as well as file permissions and authentication. It is based on GFS (Google File System). It is used to scale a single cluster node to hundreds and thousands of nodes. It handles large datasets running on commodity hardware. HDFS is highly fault-tolerant. It stores files across multiple machines. These files are stored in redundant fashion to allow for data recovery in case of failure.

PICTURE THIS...

An e-commerce website stores millions of customers' data in a distributed manner. Data has been collected over 4–5 years. It then runs batch analytics on the archived data to analyze customer's behavior, buying patterns, their preferences, their requirements, etc. This helps to understand which products are purchased by customers in which months, etc.

HBase

It stores data in HDFS. It is the first non-batch component of the Hadoop Ecosystem. It is a database on top of HDFS. It provides a quick random access to the stored data. It has very low latency compared to HDFS. It is a NoSQL database, is non-relational and is a column-oriented database. A table can have thousands of columns. A table can have multiple rows. Each row can have several column families. Each column family can have several columns. Each column can have several key values. It is based on Google BigTable. This is widely used by Facebook, Twitter, Yahoo, etc.

PICTURE THIS...

The same e-commerce website as in the HDFS case above also stores millions of product data. To search for a product among millions of products and to produce the result immediately (or you can say in real time), it needs to optimize the request and search process. HBase supports real-time analytics.

Given the huge velocity of data, they opted for HBase over HDFS, as HDFS does not support real-time writes. The results were overwhelming; it reduced the query time from 3 days to 3 minutes.

Difference between HBase and Hadoop/HDFS

1. HDFS is the file system whereas HBase is a Hadoop database. It is like NTFS and MySQL.
2. HDFS is WORM (Write once and read multiple times or many times). Latest versions support appending of data but this feature is rarely used. However, HBase supports real-time random read and write.
3. HDFS is based on Google File System (GFS) whereas HBase is based on Google Big Table.
4. HDFS supports only full table scan or partition table scan. Hbase supports random small range scan or table scan.
5. Performance of Hive on HDFS is relatively very good but for HBase it becomes 4–5 times slower.
6. The access to data is via MapReduce job only in HDFS whereas in HBase the access is via Java APIs, Rest, Avro, Thrift APIs.
7. HDFS does not support dynamic storage owing to its rigid structure whereas HBase supports dynamic storage.
8. HDFS has high latency operations whereas HBase has low latency operations.
9. HDFS is most suitable for batch analytics whereas HBase is for real-time analytics.

Hadoop Ecosystem Components for Data Ingestion

1. **Sqoop:** Sqoop stands for SQL to Hadoop. Its main functions are
   a) Importing data from RDBMS such as MySQL, Oracle, DB2, etc. to Hadoop file system (HDFS, HBase, Hive).
   b) Exporting data from Hadoop File system (HDFS, HBase, Hive) to RDBMS (MySQL, Oracle, DB2).

   Uses of Sqoop

   a) It has a connector-based architecture to allow plug-ins to connect to external systems such as MySQL, Oracle, DB2, etc.

b) It can provision the data from external system on to HDFS and populate tables in Hive and HBase.
c) It integrates with Oozie allowing you to schedule and automate import and export tasks.

2. **Flume:** Flume is an important log aggregator (aggregates logs from different machines and places them in HDFS) component in the Hadoop ecosystem. Flume has been developed by Cloudera. It is designed for high volume ingestion of event-based data into Hadoop. The default destination in Flume (called as sink in flume parlance) is HDFS. However it can also write to HBase or Solr.

PICTURE THIS...

There is a bank of web servers. Flume moves log events from those files into new aggregated files in HDFS for processing.

Hadoop Ecosystem Components for Data Processing

1. **MapReduce:** It is a programing paradigm that allows distributed and parallel processing of huge datasets. It is based on Google MapReduce. Google released a paper on MapReduce programming paradigm in 2004 and that became the genesis of Hadoop processing model. The MapReduce framework gets the input data from HDFS. There are two main phases: Map phase and the Reduce phase. The map phase converts the input data into another set of data (key–value pairs). This new intermediate dataset then serves as the input to the reduce phase. The reduce phase acts on the datasets to combine (aggregate and consolidate) and reduce them to a smaller set of tuples. The result is then stored back in HDFS.
2. **Spark:** It is both a programming model as well as a computing model. It is an open-source big data processing framework. It was originally developed in 2009 at UC Berkeley's AmpLab and became an open-source project in 2010. It is written in Scala. It provides in-memory computing for Hadoop. In Spark, workloads execute in memory rather than on disk owing to which it is much faster (10 to 100 times) than when the workload is executed on disk. However, if the datasets are too large to fit into the available system memory, it can perform conventional disk-based processing. It serves as a potentially faster and more flexible alternative to MapReduce. It accesses data from HDFS (Spark does not have its own distributed file system) but bypasses the MapReduce processing.

   Spark can be used with Hadoop coexisting smoothly with MapReduce (sitting on top of Hadoop YARN) or used independently of Hadoop (standalone). As a programming model, it works well with Scala, Python (it has API connectors for using it with Java or Python) or R programming language.

   The following are the Spark libraries:

   a) ***Spark SQL:*** Spark also has support for SQL. Spark SQL uses SQL to help query data stored in disparate applications.
   b) ***Spark streaming:*** It helps to analyze and present data in real time.
   c) ***MLib:*** It supports machine learning such as applying advanced statistical operations on data in Spark Cluster.
   d) ***GraphX:*** It helps in graph parallel computation.

   Spark and Hadoop are usually used together by several companies. Hadoop was primarily designed to house unstructured data and run batch processing operations on it. Spark is used extensively for its

high speed in memory computing and ability to run advanced real-time analytics. The two together have been giving very good results.

Hadoop Ecosystem Components for Data Analysis

1. **Pig:** It is a high-level scripting language used with Hadoop. It serves as an alternative to MapReduce. It has two parts:
   (a) ***Pig Latin:*** It is SQL-like scripting language. Pig Latin scripts are translated into MapReduce jobs which can then run on YARN and process data in the HDFS cluster. It was initially developed by Yahoo. It is immensely popular with developers who are not comfortable with MapReduce. However, SQL developers may have a preference for Hive.

   How it works? There is a "Load" command available to load the data from "HDFS" into Pig. Then one can perform functions such as grouping, filtering, sorting, joining etc. The processed or computed data can then be either displayed on screen or placed back into HDFS.

   It gives you a platform for building data flow for ETL (Extract, Transform and Load), processing and analyzing huge data sets.

   (b) ***Pig runtime:*** It is the runtime environment.
2. **Hive:** Hive is a data warehouse software project built on top of Hadoop. Three main tasks performed by Hive are summarization, querying and analysis. It supports queries written in a language called HQL or HiveQL which is a declarative SQL-like language. It converts the SQL-style queries into MapReduce jobs which are then executed on the Hadoop platform.

Difference between Hive and RDBMS

Both Hive and traditional databases such as MySQL, MS SQL Server, PostgreSQL support SQL interface. However, Hive is better known as a datawarehouse (D/W) rather than a database.

Let us look at the difference between Hive and traditional databases as regards the schema.

1. Hive enforces schema on Read Time whereas RDBMS enforces schema on Write Time. In RDBMS, at the time of loading/inserting data, the table's schema is enforced. If the data being loaded does not conform to the schema then it is rejected. Thus, the schema is enforced on write (loading the data into the database). Schema on write takes longer to load the data into the database; however it makes up for it during data retrieval with a good query time performance. However, Hive does not enforce the schema when the data is being loaded into the D/W. It is enforced only when the data is being read/retrieved. This is called schema on read. It definitely makes for fast initial load as the data load or insertion operation is just a file copy or move.
2. Hive is based on the notion of write once and read many times whereas the RDBMS is designed for read and write many times.
3. Hadoop is a batch-oriented system. Hive, therefore, is not suitable for OLTP (Online Transaction Processing) but, although not ideal, seems closer to OLAP (Online Analytical Processing). The reason being that there is quite a latency between issuing a query and receiving a reply as the query written in HiveQL will be converted to MapReduce jobs which are then executed on the Hadoop cluster. RDBMS is suitable for housing day-to-day transaction data and supports all OLTP operations with frequent insertions, modifications (updates), deletions of the data.

4. Hive handles static data analysis which is non-real-time data. Hive is the data warehouse of Hadoop. There are no frequent updates to the data and the query response time is not fast. RDBMS is suited for handling dynamic data which is real time.
5. Hive can be easily scaled at a very low cost when compared to RDMS. Hive uses HDFS to store data, thus it cannot be considered as the owner of the data, while on the other hand RDBMS is the owner of the data responsible for storing, managing and manipulating it in the database.
6. Hive uses the concept of parallel computing, whereas RDBMS uses serial computing.

We summarize the difference in Table 4.5.

Table 4.5 Hive versus RDBMS

	Hadoop	RDBMS
Data Variety	Used for structured, semi-structured and unstructured data. Hadoop supports a variety of data formats in real time such as XML, JSON, and text-based flat file formats.	Used for structured data
Data Storage	Usually datasets of size terabytes, petabytes	Usually datasets of size gigabytes
Querying	HiveQL	SQL
Query Response	In Hadoop, there is latency due to batch processing.	In RDBMS, query response time is immediate.
Schema	Schema required on read	Schema required on write
Speed	Writes are faster compared to reads as there is no adherence to schema required at the time of inserting or writing data. Schema is enforced at read time	Reads are very fast (supported by building indexes on required columns).
	Hadoop is designed for write once read many times. It does not work for random reading and writing of a few records like RDBMS.	RDBMS is designed for read and write many times.
Cost	Apache Hadoop is open-source, large-scale, distributed, scalable, data intensive computing.	Available as proprietary RDBMS such as Oracle, MS SQL Server, IBM DB2, etc. Also open-source RDBMS are available such as MySQL, PostgreSQL, etc.
Use Cases	Analytics, data discovery	OLTP (Online Transaction Processing). Mainly used to store and process day-to-day business data.

(Continued)

Table 4.5 (Continued)

	Hadoop	RDBMS
Throughput	High	Low
Scalability	Horizontal (Hadoop scales by adding nodes to a Hadoop cluster of easily available commodity machines).	Vertical: RDBMS scales vertically by increasing the horsepower (CPU, Hard Disk Capacity, RAM, etc.) of the machine.
Hardware	Commodity/Utility Hardware	High End Servers
Integrity	Low	High. Obeys ACID properties A – Atomicity C – Consistency I – Integrity D – Durability

Difference between Hive and HBase

1. Hive is a MapReduce-based SQL engine that runs on top of Hadoop. HBase is a key–value NoSQL database that runs on top of HDFS.
2. Hive is for batch processing of big data. HBase is for real-time data streaming.

Impala

It is a high performance SQL engine that runs on Hadoop cluster. It is ideal for interactive analysis. It has very low latency measured in milliseconds. It supports a dialect of SQL called Impala SQL.

ZooKeeper

It is a coordination service for distributed applications.

Oozie

It is a workflow scheduler system to manage Apache Hadoop jobs.

Mahout

It is a scalable machine learning and data mining library.

Chukwa

It is a data collection system for managing large distributed systems.

Ambari

It is a web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters.

4.2.5 Hadoop Distributions

Hadoop is an open-source Apache project. Anyone can freely download the core aspects of Hadoop. The core aspects of Hadoop include the following:

1. Hadoop Common
2. **H**adoop **D**istributed **F**ile **S**ystem (HDFS)
3. Hadoop YARN (**Y**et **A**nother **R**esource **N**egotiator)
4. Hadoop MapReduce

There are few companies such as IBM, Amazon Web Services, Microsoft, Teradata, Hortonworks, Cloudera, etc. that have packaged Hadoop into a more easily consumable distributions or services. Although each of these companies have a slightly different strategy, the key essence remains its ability to distribute data and workloads across potentially thousands of servers thus making big data manageable data. A few Hadoop distributions are given in Figure 4.12.

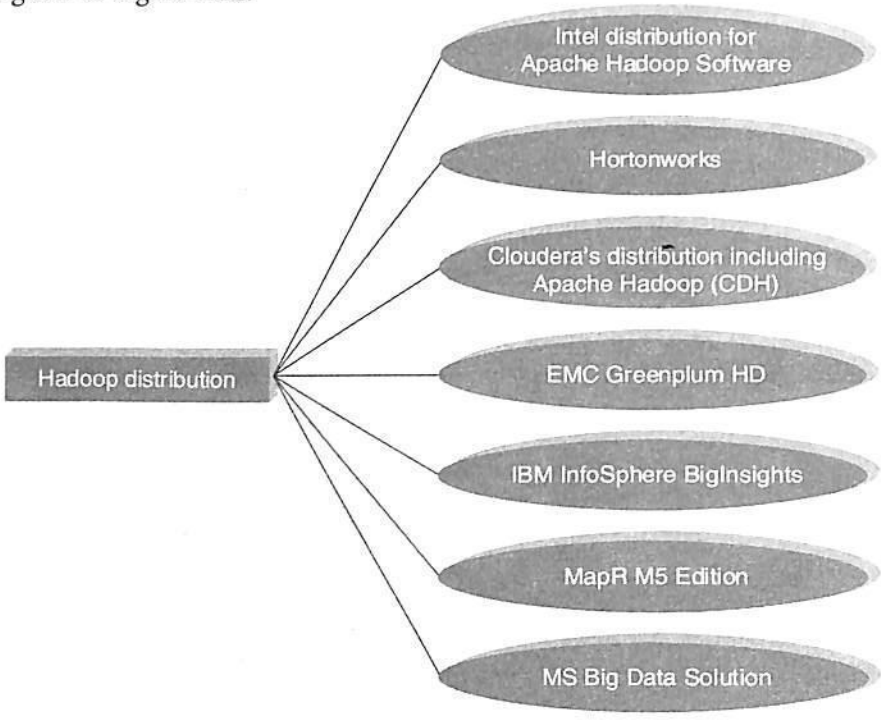


Figure 4.12 Hadoop distributions.

4.2.6 Hadoop versus SQL

Table 4.6 lists the differences between Hadoop and SQL.

Table 4.6 Hadoop versus SQL

Hadoop	SQL
Scale out	Scale up
Key–Value pairs	Relational table
Functional Programming	Declarative Queries
Offline batch processing	Online transaction processing

4.2.7 Integrated Hadoop Systems Offered by Leading Market Vendors

Refer Figure 4.13 to get a glimpse of the leading market vendors offering integrated Hadoop systems.

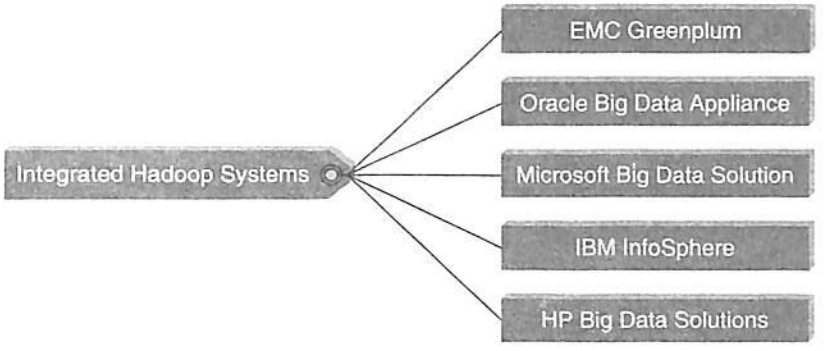


Figure 4.13 Integrated Hadoop systems.

4.2.8 Cloud-Based Hadoop Solutions

Amazon Web Services holds out a comprehensive, end-to-end portfolio of cloud computing services to help manage big data. The aim is to achieve this and more along with retaining the emphasis on reducing costs, scaling to meet demand, and accelerating the speed of innovation.

The Google Cloud Storage connector for Hadoop empowers one to perform MapReduce jobs directly on data in Google Cloud Storage, without the need to copy it to local disk and running it in the Hadoop Distributed File System (HDFS). The connector simplifies Hadoop deployment, and at the same time reduces cost and provides performance comparable to HDFS, all this while increasing reliability by eliminating the single point of failure of the name node. Refer Figure 4.14.

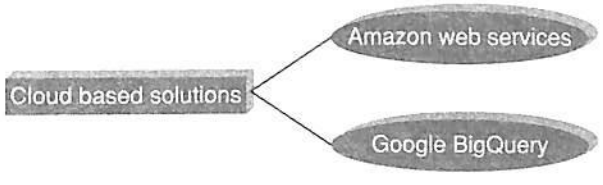


Figure 4.14 Cloud-based solutions.

REMIND ME

- NoSQL databases are non-relational, open source, distributed databases.
- NoSQL database allows insertion of data without a pre-defined schema.
- Hadoop has a shared nothing architecture.

- Hadoop 1.0 has two main parts:
  - Data storage framework
  - Data processing framework
- In Hadoop 2.0, a new and separate resource management framework called Yet Another Resource Negotiator (YARN) has been added.

POINT ME (BOOKS)

- Hadoop for Dummies, Dirk Deroos, Paul C. Zikopoulos, Roman B. Melnyk, Bruce Brown, Wiley India Pvt. Ltd.
- NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Pramod J. Sadalage and Martin Fowler.

CONNECT ME (INTERNET RESOURCES)

- http://www.mongodb.com/nosql-explained
- http://nosql-database.org/
- http://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-nosql-databases/
- http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduce_Compatibility_Hadoop1_Hadoop2.html
- http://hadoop.apache.org/

TEST ME

A. Fill Me

1. The expansion for CAP is _____, _____ and _____.
2. The expansion of BASE is _____.
3. MongoDB is _____ and _____.
4. Cassandra is _____ and _____.
5. _____ has no support for ACID properties of transactions.
6. _____ is a robust database that supports ACID properties of transactions and has the scalability of NoSQL.

Answers:

1. Consistency, Availability and Partition Tolerance
2. Basically Available Soft State Eventual Consistency
3. Consistent and Partition Tolerant
4. Available and Partition Tolerant
5. NoSQL
6. NewSQL

B. Place it in the Basket

SQL	NoSQL

Following words are to be placed in the relevant basket:

(a) Relational
(b) Distributed
(c) Predefined schema
(d) Wide-column stores
(e) Vertically scalable
(f) Key–value pairs
(g) MySQL
(h) CouchDB
(i) Neo4j
(j) Cassandra
(k) Large dataset
(l) ACID properties
(m) Brewers CAP theorem
(n) Document based database
(o) Scales horizontally
(p) Avoids join operations
(q) JSON data
(r) Table or relations

Answers:

SQL	NoSQL
Relational	Distributed
Predefined schema	Wide-column stores
Vertically scalable	Key–value pairs
MySQL	CouchDB
ACID properties	Neo4j
Table or relations	Cassandra
	Large dataset
	Brewers CAP theorem
	Document based database
	Scales horizontally
	Avoids join operations
	JSON data

CHAPTER 5

Introduction to Hadoop

BRIEF CONTENTS

- What's in Store?
- Introducing Hadoop
  - Data: The Treasure Trove
- Why Hadoop?
- Why not RDBMS?
- RDBMS versus Hadoop
- Distributed Computing Challenges
  - Hardware Failure
  - How to Process this Gigantic Store of Data?
- History of Hadoop
  - The Name "Hadoop"
- Hadoop Overview
  - Key Aspects of Hadoop
  - Hadoop Components
  - Hadoop Conceptual Layer
  - High-Level Architecture of Hadoop
- Use Case for Hadoop
  - ClickStream Data
- Hadoop Distributors
- HDFS
  - HDFS Daemons
  - Anatomy of File Read
  - Anatomy of File Write
  - Replica Placement Strategy
  - Working with HDFS Commands
  - Special Features of HDFS
- Processing Data with Hadoop
  - MapReduce Daemons
  - How does MapReduce Work?
  - MapReduce Example
- Managing Resources and Applications with Hadoop YARN
  - Limitations of Hadoop 1.0 Architecture
  - HDFS Limitation
  - Hadoop 2: HDFS
  - Hadoop 2 YARN: Taking Hadoop Beyond Batch
- Interacting with Hadoop Ecosystem
  - Pig
  - Hive
  - Sqoop
  - HBase

"There were 5 exabytes of information created between the dawns of civilization through 2003, but that much information is now created every 2 days."

– Eric Schmidt, of Google, said in 2010

WHAT'S IN STORE?

We assume that you are already familiar with the distributed file system and the distributed computing model. The focus of this chapter will be to build on this knowledge base and comprehend and appreciate how Hadoop stores and processes colossal volumes of data. It will be our endeavor to get you the importance of Hadoop with case studies and scenarios. We will also discuss HDFS commands and MapReduce Programming. However, MapReduce Programming will be discussed in detail in Chapter 8.

We suggest you refer to some of the learning resources provided at the end of this chapter and also complete the "Test Me" exercises.

5.1 INTRODUCING HADOOP

Today, Big Data seems to be the buzz word! Enterprises, the world over, are beginning to realize that there is a huge volume of untapped information before them in the form of structured, semi-structured, and unstructured data. This varied variety of data is spread across the networks.

Let us look at few statistics to get an idea of the amount of data which gets generated every day, every minute, and every second.

1. **Every day:**
   (a) NYSE (New York Stock Exchange) generates 1.5 billion shares and trade data.
   (b) Facebook stores 2.7 billion comments and Likes.
   (c) Google processes about 24 petabytes of data.
2. **Every minute:**
   (a) Facebook users share nearly 2.5 million pieces of content.
   (b) Twitter users tweet nearly 300,000 times.
   (c) Instagram users post nearly 220,000 new photos.
   (d) YouTube users upload 72 hours of new video content.
   (e) Apple users download nearly 50,000 apps.
   (f) Email users send over 200 million messages.
   (g) Amazon generates over $80,000 in online sales.
   (h) Google receives over 4 million search queries.
3. **Every second:**
   (a) Banking applications process more than 10,000 credit card transactions.

5.1.1 Data: The Treasure Trove

1. Provides business advantages such as generating product recommendations, inventing new products, analyzing the market, and many, many more, ...
2. Provides few early key indicators that can turn the fortune of business.
3. Provides room for precise analysis. If we have more data for analysis, then we have greater precision of analysis.

To process, analyze, and make sense of these different kinds of data, we need a system that scales and addresses the challenges shown in Figure 5.1.

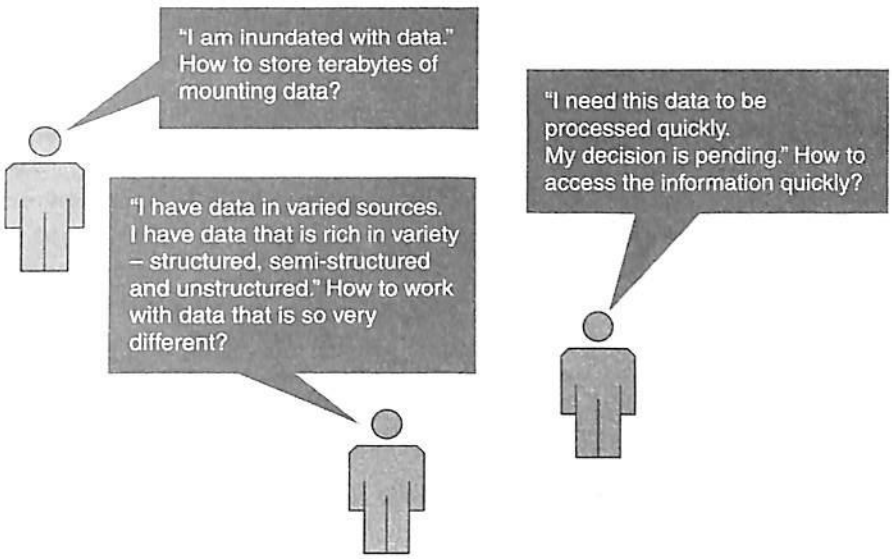


Figure 5.1 Challenges with big volume, variety, and velocity of data.

5.2 WHY HADOOP?

Ever wondered why Hadoop has been and is one of the most wanted technologies!!

The key consideration (the rationale behind its huge popularity) is:

Its capability to handle massive amounts of data, different categories of data – fairly quickly.

The other considerations are (Figure 5.2):

1. **Low cost:** Hadoop is an open-source framework and uses commodity hardware (commodity hardware is relatively inexpensive and easy to obtain hardware) to store enormous quantities of data.

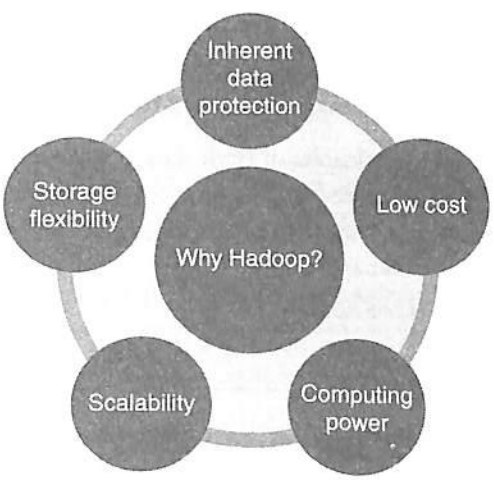


Figure 5.2 Key considerations of Hadoop.

2. **Computing power:** Hadoop is based on distributed computing model which processes very large volumes of data fairly quickly. The more the number of computing nodes, the more the processing power at hand.
3. **Scalability:** This boils down to simply adding nodes as the system grows and requires much less administration.
4. **Storage flexibility:** Unlike the traditional relational databases, in Hadoop data need not be pre-processed before storing it. Hadoop provides the convenience of storing as much data as one needs and also the added flexibility of deciding later as to how to use the stored data. In Hadoop, one can store unstructured data like images, videos, and free-form text.
5. **Inherent data protection:** Hadoop protects data and executing applications against hardware failure. If a node fails, it automatically redirects the jobs that had been assigned to this node to the other functional and available nodes and ensures that distributed computing does not fail. It goes a step further to store multiple copies (replicas) of the data on various nodes across the cluster.

Hadoop makes use of commodity hardware, distributed file system, and distributed computing as shown in Figure 5.3. In this new design, groups of machine are gathered together; it is known as a **Cluster**.

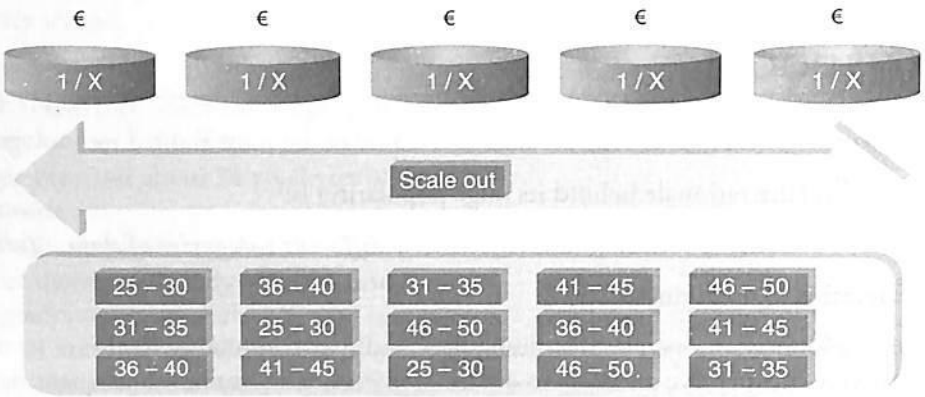


Figure 5.3 Hadoop framework (distributed file system, commodity hardware).

With this new paradigm, the data can be managed with **Hadoop** as follows:

1. Distributes the data and duplicates chunks of each data file across several nodes, for example, 25–30 is one chunk of data as shown in Figure 5.3.
2. Locally available compute resource is used to process each chunk of data in parallel.
3. Hadoop Framework handles failover smartly and automatically.

5.3 WHY NOT RDBMS?

RDBMS is not suitable for storing and processing large files, images, and videos. RDBMS is not a good choice when it comes to advanced analytics involving machine learning. Figure 5.4 describes the RDBMS system with respect to cost and storage. It calls for huge investment as the volume of data shows an upward trend.

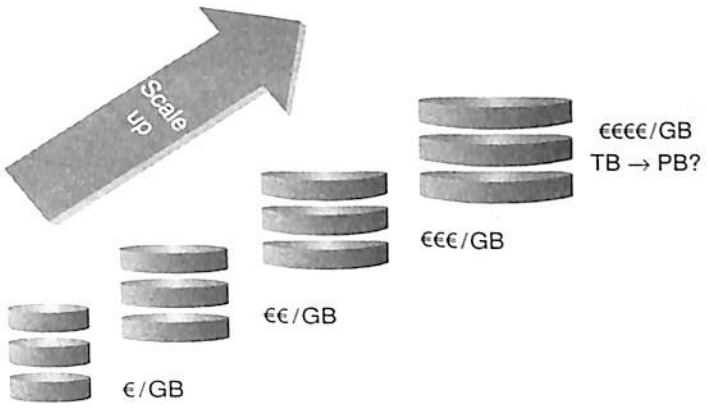


Figure 5.4 RDBMS with respect to cost/GB of storage.

5.4 RDBMS versus HADOOP

Table 5.1 describes the difference between RDBMS and Hadoop.

Table 5.1 RDBMS versus Hadoop

PARAMETERS	RDBMS	HADOOP
System	Relational Database Management System.	Node Based Flat Structure.
Data	Suitable for structured data.	Suitable for structured, unstructured data. Supports variety of data formats in real time such as XML, JSON, text based flat file formats, etc.
Processing	OLTP	Analytical, Big Data Processing
Choice	When the data needs consistent relationship.	Big Data processing, which does not require any consistent relationships between data.
Processor	Needs expensive hardware or high-end processors to store huge volumes of data.	In a Hadoop Cluster, a node requires only a processor, a network card, and few hard drives.
Cost	Cost around $10,000 to $14,000 per terabytes of storage.	Cost around $4,000 per terabytes of storage.

5.5 DISTRIBUTED COMPUTING CHALLENGES

Although there are several challenges with distributed computing, we will focus on two major challenges.

5.5.1 Hardware Failure

In a distributed system, several servers are networked together. This implies that more often than not, there may be a possibility of hardware failure. And when such a failure does happen, how does one retrieve the

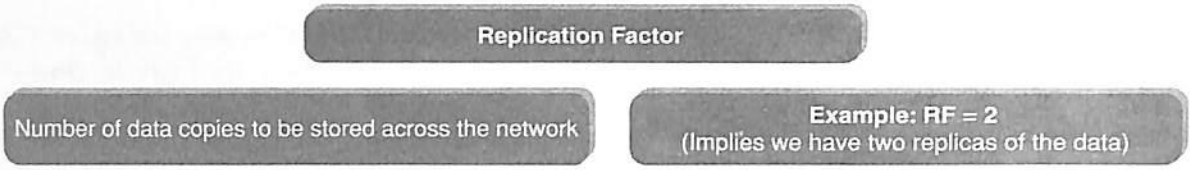


Figure 5.5 Replication factor.

data that was stored in the system? Just to explain further – a regular hard disk may fail once in 3 years. And when you have 1000 such hard disks, there is a possibility of at least a few being down every day.

Hadoop has an answer to this problem in **Replication Factor (RF)**. **Replication Factor** connotes the number of data copies of a given data item/data block stored across the network. Refer Figure 5.5.

JUST TO UNDERSTAND REPLICATION FURTHER, PICTURE THIS...

You work in a project team. There are six other members in the team. Each time there is an update related to the project work or an input received from the client, the project manager, Alex, ensures that he keeps at least three team members aware of the developments. You have been wondering at this style of working of your project manager. One day during the coffee break, when the project manager joins for coffee, you hesitantly ask him the question. Alex, "I had this question for you. Why is that each time we have an input from the client or any important piece of information, you leave it with at least three of our team members?" Alex smiled as he answered, "The reason is very simple. Assume that the client called and suggested some modification to the project. I shared it with just one person, let us say, person X. Tomorrow, when the suggested changes have to be incorporated, person X calls in sick. He is indisposed and not in office. Will that lead to our project coming to a standstill? Yes, isn't it? Therefore I share it with at least three team members, so that even if one is on leave or out of office for some reason, our work will not be stalled."

5.5.2 How to Process This Gigantic Store of Data?

In a distributed system, the data is spread across the network on several machines. A key challenge here is to integrate the data available on several machines prior to processing it.

Hadoop solves this problem by using **MapReduce** Programming. It is a programming model to process the data (MapReduce programming will be discussed a little later).

5.6 HISTORY OF HADOOP

Hadoop was created by Doug Cutting, the creator of Apache Lucene (a commonly used text search library). Hadoop is a part of the Apache Nutch (Yahoo) project (an open-source web search engine) and also a part of the Lucene project. Refer Figure 5.6 for more details.

5.6.1 The Name "Hadoop"

The name Hadoop is not an acronym; it's a made-up name. The project creator, Doug Cutting, explains how the name came about: "*The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere: those are my naming criteria. Kids are good at generating such. Googol is a kid's term*".

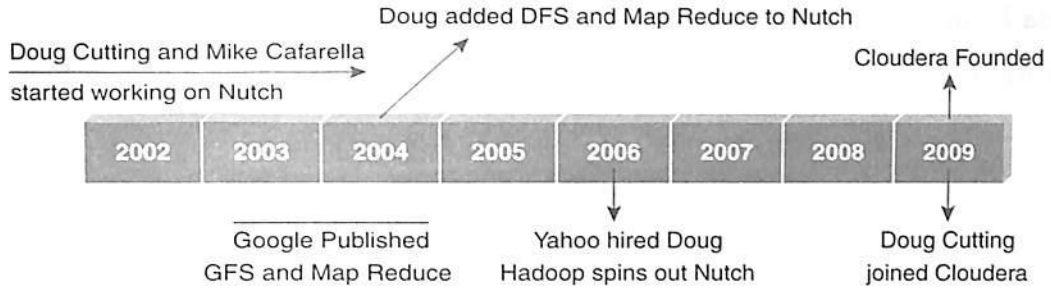


Figure 5.6 Hadoop history.

Subprojects and "contrib" modules in Hadoop also tend to have names that are unrelated to their function, often with an elephant or other animal theme ("Pig", for example).

Reference: Hadoop, The Definitive Guide, 3rd Edition, O'Reilly Publication Page. No. 9.

5.7 HADOOP OVERVIEW

Open-source software framework to store and process massive amounts of data in a distributed fashion on large clusters of commodity hardware. Basically, Hadoop accomplishes two tasks:

1. Massive data storage.
2. Faster data processing.

5.7.1 Key Aspects of Hadoop

Figure 5.7 describes the key aspects of Hadoop.

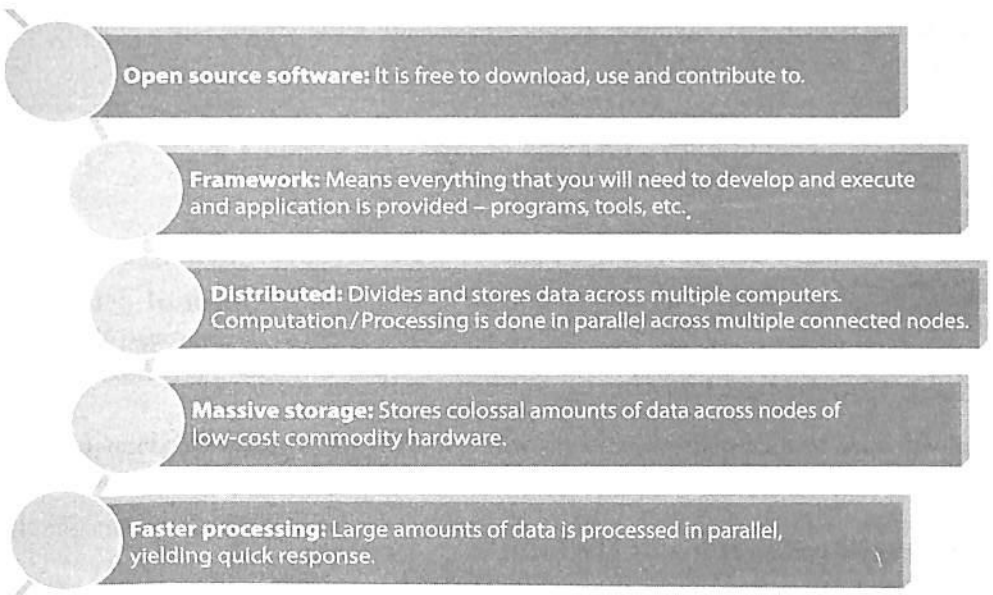


Figure 5.7 Key aspects of Hadoop.

5.7.2 Hadoop Components

Figure 5.8 depicts the Hadoop components.

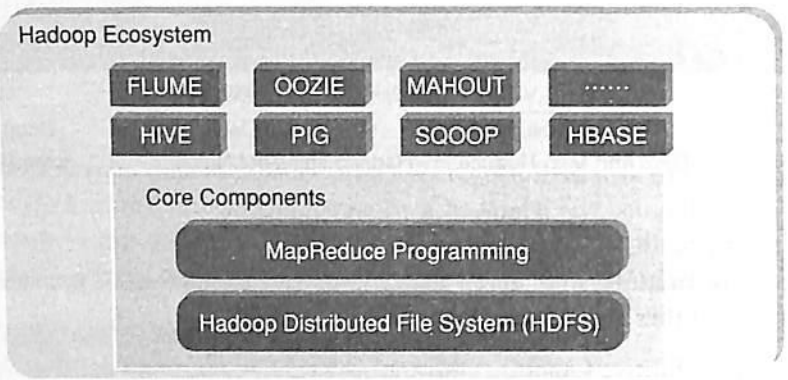


Figure 5.8 Hadoop components.

Hadoop Core Components

1. **HDFS:**
   (a) Storage component.
   (b) Distributes data across several nodes.
   (c) Natively redundant.
2. **MapReduce:**
   (a) Computational framework.
   (b) Splits a task across multiple nodes.
   (c) Processes data in parallel.

Hadoop Ecosystem: Hadoop Ecosystem are support projects to enhance the functionality of Hadoop Core Components. The Eco Projects are as follows:

1. HIVE
2. PIG
3. SQOOP
4. HBASE
5. FLUME
6. OOZIE
7. MAHOUT

5.7.3 Hadoop Conceptual Layer

It is conceptually divided into **Data Storage Layer** which stores huge volumes of data and **Data Processing Layer** which processes data in parallel to extract richer and meaningful insights from data (Figure 5.9).

5.7.4 High-Level Architecture of Hadoop

Hadoop is a distributed **Master-Slave** Architecture. Master node is known as **NameNode** and slave nodes are known as **DataNodes.** Figure 5.10 depicts the Master–Slave Architecture of Hadoop Framework.

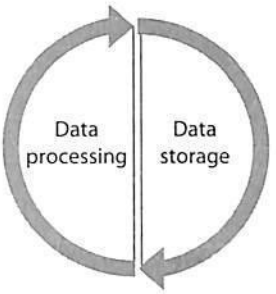


Figure 5.9 Hadoop conceptual layer.

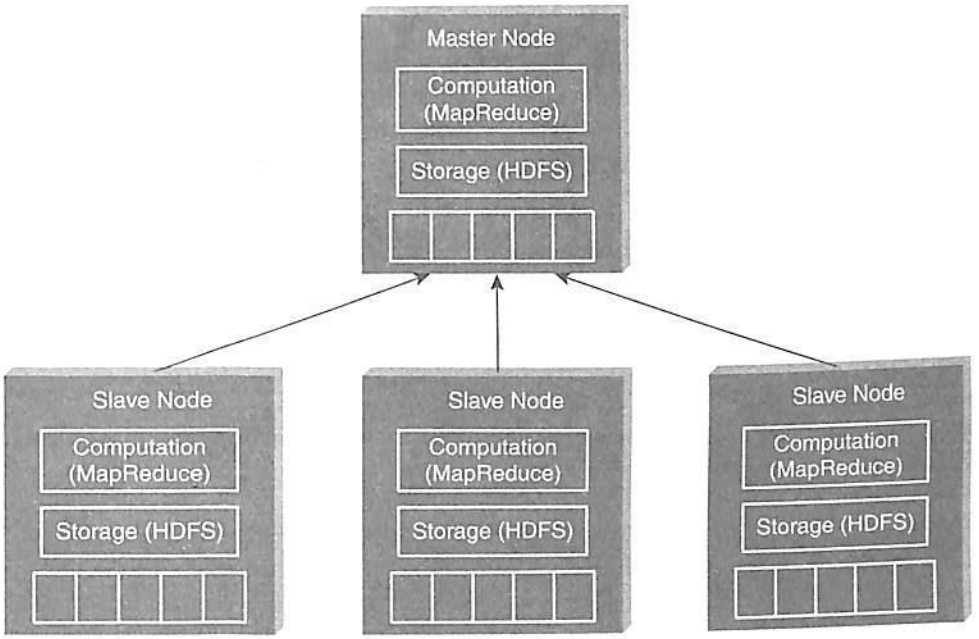


Figure 5.10 Hadoop high-level architecture.
Reference: Hadoop in Practice, Alex Holmes.

Let us look at the key components of the Master Node.

1. **Master HDFS:** Its main responsibility is partitioning the data storage across the slave nodes. It also keeps track of locations of data on DataNodes.
2. **Master MapReduce:** It decides and schedules computation task on slave nodes.

5.8 USE CASE OF HADOOP

5.8.1 ClickStream Data

ClickStream data (mouse clicks) helps you to understand the purchasing behavior of customers. ClickStream analysis helps online marketers to optimize their product web pages, promotional content, etc. to improve their business.

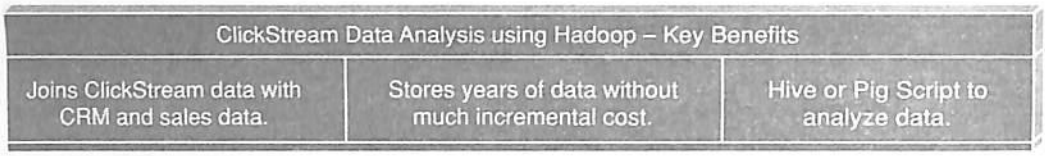


Figure 5.11 ClickStream data analysis.

The ClickStream analysis (Figure 5.11) using Hadoop provides **three key benefits**:

1. Hadoop helps to join ClickStream data with other data sources such as Customer Relationship Management Data (Customer Demographics Data, Sales Data, and Information on Advertising Campaigns). This additional data often provides the much needed information to understand customer behavior.
2. Hadoop's scalability property helps you to store years of data without ample incremental cost. This helps you to perform temporal or year over year analysis on ClickStream data which your competitors may miss.
3. Business analysts can use **Apache Pig** or **Apache Hive** for website analysis. With these tools, you can organize ClickStream data by user session, refine it, and feed it to visualization or analytics tools.

Reference: http://hortonworks.com/wp-content/uploads/2014/05/Hortonworks.BusinessValueofHadoop.v1.0.pdf

5.9 HADOOP DISTRIBUTORS

The companies shown in Figure 5.12 provide products that include Apache Hadoop, commercial support, and/or tools and utilities related to Hadoop.

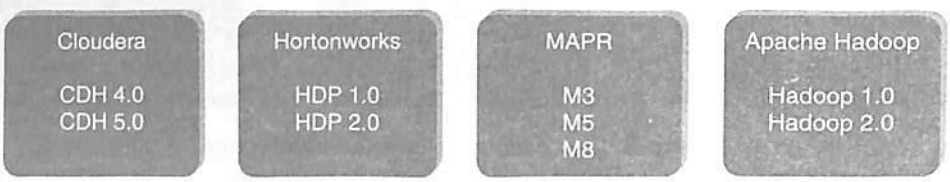


Figure 5.12 Common Hadoop distributors.

5.10 HDFS (HADOOP DISTRIBUTED FILE SYSTEM)

Some key Points of Hadoop Distributed File System are as follows:

1. Storage component of Hadoop.
2. Distributed File System.
3. Modeled after Google File System.
4. Optimized for high throughput (HDFS leverages large block size and moves computation where data is stored).
5. You can replicate a file for a configured number of times, which is tolerant in terms of both software and hardware.

6. Re-replicates data blocks automatically on nodes that have failed.
7. You can realize the power of HDFS when you perform read or write on large files (gigabytes and larger).
8. Sits on top of native file system such as ext3 and ext4, which is described in Figure 5.13.

Figure 5.14 describes important key points of HDFS. Figure 5.15 describes Hadoop Distributed File System Architecture. Client Application interacts with NameNode for metadata related activities and communicates with DataNodes to read and write files. DataNodes converse with each other for pipeline reads and writes.

Let us assume that the file "Sample.txt" is of size **192 MB**. As per the default data block size (64 MB), it will be split into three blocks and replicated across the nodes on the cluster based on the default replication factor.

5.10.1 HDFS Daemons

5.10.1.1 NameNode

HDFS breaks a large file into smaller pieces called **blocks**. NameNode uses a **rack ID** to identify DataNodes in the rack. A rack is a collection of DataNodes within the cluster. NameNode keeps tracks of blocks of a file as it is placed on various DataNodes. NameNode manages file-related operations such as read, write, create, and delete. Its main job is managing the **File System Namespace**. A file system namespace is collection of files in the cluster. NameNode stores HDFS namespace. File system namespace includes mapping of blocks to file, file properties and is stored in a file called **FsImage**. NameNode uses an EditLog (transaction log) to record every transaction that happens to the file system metadata. Refer Figure 5.16. When NameNode starts up, it reads FsImage and EditLog from disk and applies all transactions from the EditLog to in-memory representation of the FsImage. Then it flushes out new version of FsImage on disk and truncates the old EditLog because the changes are updated in the FsImage. There is a single NameNode per cluster.

Reference: http://hadoop.apache.org/docs/r1.0.4/hdfs_design.html

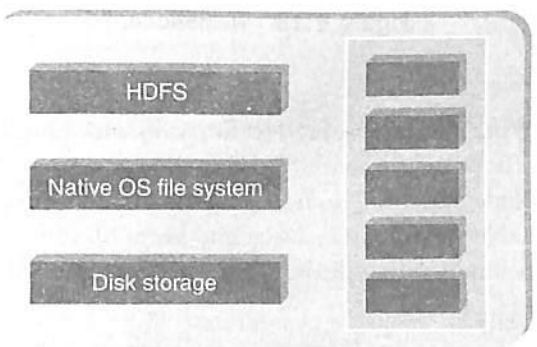


Figure 5.13 Hadoop Distributed File System.

Figure 5.14 Hadoop Distributed File System – key points.

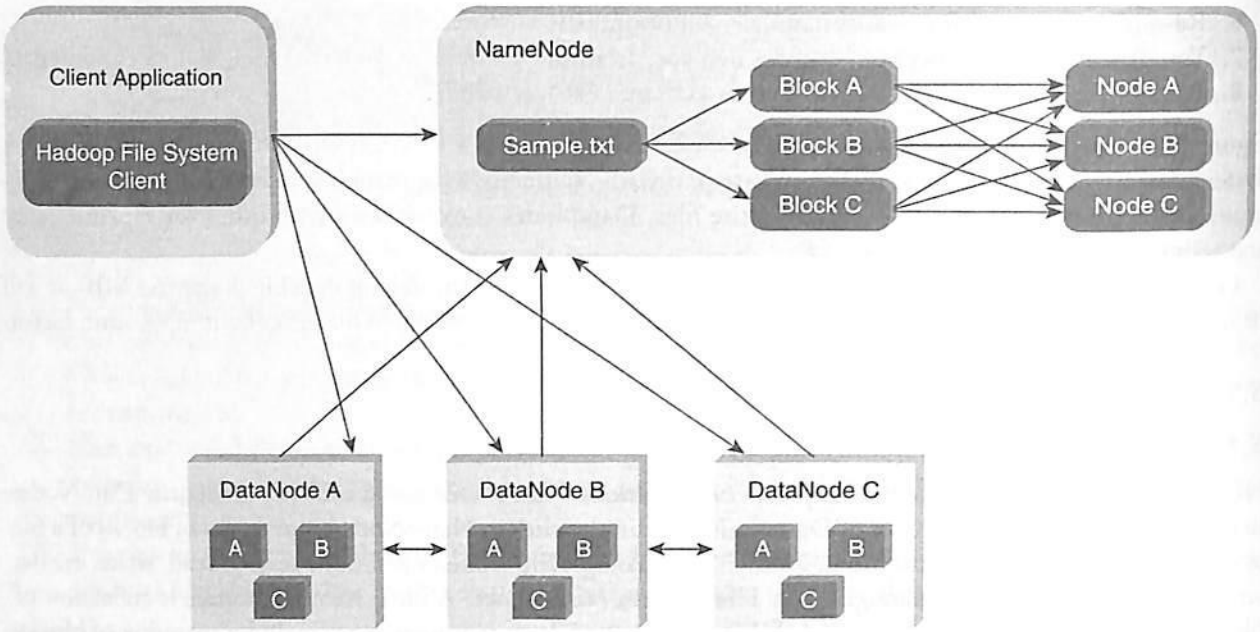


Figure 5.15 Hadoop Distributed File System Architecture.
Reference: Hadoop in Practice, Alex Holmes.

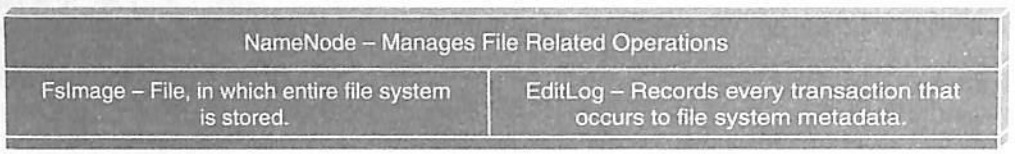


Figure 5.16 NameNode.

5.10.1.2 DataNode

There are multiple DataNodes per cluster. During Pipeline read and write DataNodes communicate with each other. A DataNode also continuously sends **"heartbeat"** message to NameNode to ensure the connectivity between the NameNode and DataNode. In case there is no heartbeat from a DataNode, the NameNode replicates that DataNode within the cluster and keeps on running as if nothing had happened.

Let us explain the concept behind sending the heartbeat report by the DataNodes to the NameNode.

Reference: Wrox Certified Big Data Developer.

PICTURE THIS...

You work for a renowned IT organization. Every day when you come to office, you are required to swipe in to record your attendance. This record of attendance is then shared with your manager to keep him posted on who all from his team have reported for work. Your manager is able to allocate tasks to the team members who are present in office. The tasks for the day cannot be allocated to team members who have not turned in. Likewise heartbeat report is a way by which DataNodes inform the NameNode that they are up and functional and can be assigned tasks. Figure 5.17 depicts the above scenario.

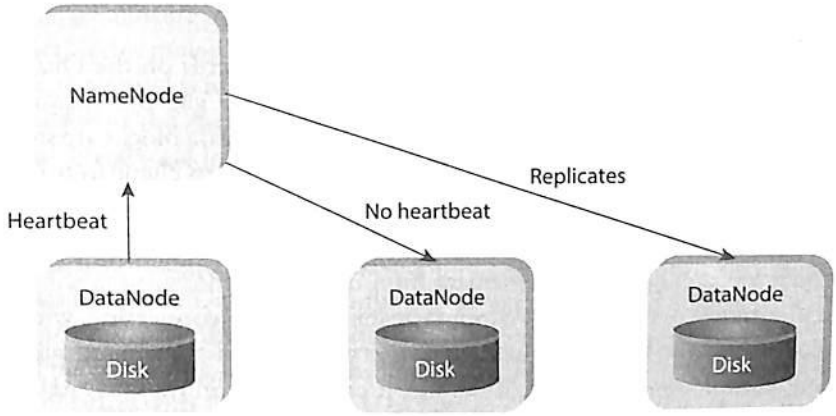


Figure 5.17 NameNode and DataNode Communication.

5.10.1.3 *Secondary NameNode*

The Secondary NameNode takes a snapshot of HDFS metadata at intervals specified in the Hadoop configuration. Since the memory requirements of Secondary NameNode are the same as NameNode, it is better to run NameNode and Secondary NameNode on different machines. In case of failure of the NameNode, the Secondary NameNode can be configured manually to bring up the cluster. However, the Secondary NameNode does not record any real-time changes that happen to the HDFS metadata.

5.10.2 Anatomy of File Read

Figure 5.18 describes the anatomy of File Read.

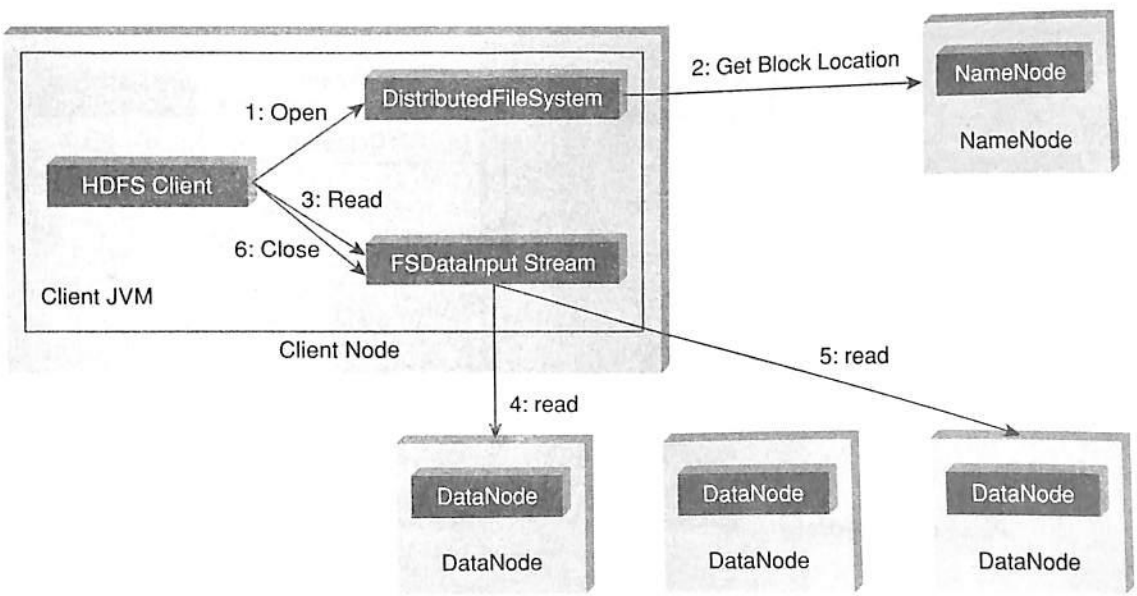


Figure 5.18 File Read.

The steps involved in the File Read are as follows:

1. The client opens the file that it wishes to read from by calling open() on the DistributedFileSystem.
2. DistributedFileSystem communicates with the NameNode to get the location of data blocks. NameNode returns with the addresses of the DataNodes that the data blocks are stored on. Subsequent to this, the DistributedFileSystem returns an FSDataInputStream to client to read from the file.
3. Client then calls read() on the stream DFSInputStream, which has addresses of the DataNodes for the first few blocks of the file, connects to the closest DataNode for the first block in the file.
4. Client calls read() repeatedly to stream the data from the DataNode.
5. When end of the block is reached, DFSInputStream closes the connection with the DataNode. It repeats the steps to find the best DataNode for the next block and subsequent blocks.
6. When the client completes the reading of the file, it calls close() on the FSDataInputStream to close the connection.

Reference: Hadoop, The Definitive Guide, 3rd Edition, O'Reilly Publication.

5.10.3 Anatomy of File Write

Figure 5.19 describes the anatomy of File Write. The steps involved in anatomy of File Write are as follows:

1. The client calls create() on DistributedFileSystem to create a file.
2. An RPC call to the NameNode happens through the DistributedFileSystem to create a new file. The NameNode performs various checks to create a new file (checks whether such a file exists or not). Initially, the NameNode creates a file without associating any data blocks to the file. The DistributedFileSystem returns an FSDataOutputStream to the client to perform write.
3. As the client writes data, data is split into packets by DFSOutputStream, which is then written to an internal queue, called *data queue*. DataStreamer consumes the data queue. The DataStreamer requests

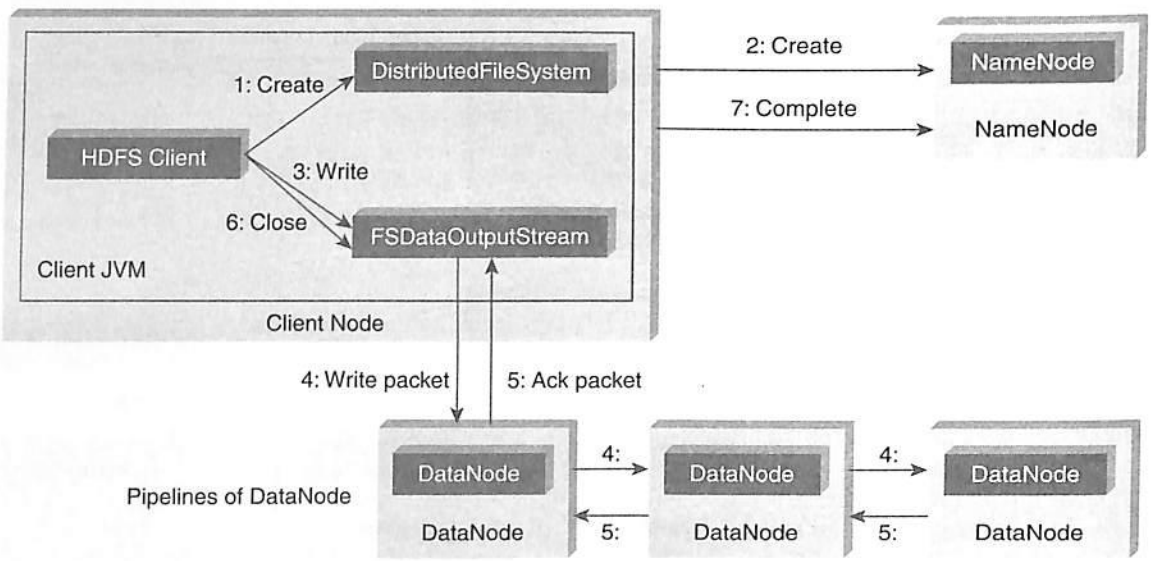


Figure 5.19 File Write.

the NameNode to allocate new blocks by selecting a list of suitable DataNodes to store replicas. This list of DataNodes makes a pipeline. Here, we will go with the default replication factor of three, so there will be three nodes in the pipeline for the first block.

4. DataStreamer streams the packets to the first DataNode in the pipeline. It stores packet and forwards it to the second DataNode in the pipeline. In the same way, the second DataNode stores the packet and forwards it to the third DataNode in the pipeline.
5. In addition to the internal queue, DFSOutputStream also manages an "Ack queue" of packets that are waiting for the acknowledgement by DataNodes. A packet is removed from the "Ack queue" only if it is acknowledged by all the DataNodes in the pipeline.
6. When the client finishes writing the file, it calls close() on the stream.
7. This flushes all the remaining packets to the DataNode pipeline and waits for relevant acknowledgments before communicating with the NameNode to inform the client that the creation of the file is complete.

Reference: Hadoop, The Definitive Guide, 3rd Edition, O'Reilly Publication.

5.10.4 Replica Placement Strategy

5.10.4.1 *Hadoop Default Replica Placement Strategy*

As per the Hadoop Replica Placement Strategy, first replica is placed on the same node as the client. Then it places second replica on a node that is present on different rack. It places the third replica on the same rack as second, but on a different node in the rack. Once replica locations have been set, a pipeline is built. This strategy provides good reliability. Figure 5.20 describes the typical replica pipeline.

Reference: Hadoop, the Definite Guide, 3rd Edition, O'Reilly Publication.

5.10.5 Working with HDFS Commands

Objective: To get the list of directories and files at the root of HDFS.
Act:
hadoop fs -ls /

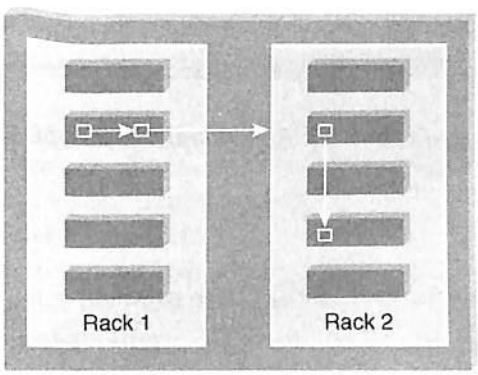


Figure 5.20 Replica Placement Strategy.

Objective: To get the list of complete directories and files of HDFS.
Act:
hadoop fs -ls -R /

Objective: To create a directory (say, sample) in HDFS.
Act:
hadoop fs -mkdir /sample

Objective: To copy a file from local file system to HDFS.
Act:
hadoop fs -put /root/sample/test.txt /sample/test.txt

Objective: To copy a file from HDFS to local file system.
Act:
hadoop fs -get /sample/test.txt /root/sample/testsample.txt

Objective: To copy a file from local file system to HDFS via copyFromLocal command.
Act:
hadoop fs -copyFromLocal /root/sample/test.txt /sample/testsample.txt

Objective: To copy a file from Hadoop file system to local file system via copyToLocal command.
Act:
hadoop fs -copyToLocal /sample/test.txt /root/sample/testsample1.txt

Objective: To display the contents of an HDFS file on console.
Act:
hadoop fs -cat /sample/test.txt

Objective: To copy a file from one directory to another on HDFS.
Act:
hadoop fs -cp /sample/test.txt /sample1

Objective: To remove a directory from HDFS.
Act:
hadoop fs-rm-r /sample1

5.10.6 Special Features of HDFS

1. **Data Replication:** There is absolutely no need for a client application to track all blocks. It directs the client to the nearest replica to ensure high performance.
2. **Data Pipeline:** A client application writes a block to the first DataNode in the pipeline. Then this DataNode takes over and forwards the data to the next node in the pipeline. This process continues for all the data blocks, and subsequently all the replicas are written to the disk.

Reference: Wrox Certified Big Data Developer.

5.11 PROCESSING DATA WITH HADOOP

MapReduce Programming is a software framework. MapReduce Programming helps you to process massive amounts of data in parallel.

In MapReduce Programming, the input dataset is split into independent chunks. **Map tasks** process these independent chunks completely in a parallel manner. The output produced by the map tasks serves as intermediate data and is stored on the local disk of that server. The output of the mappers are automatically shuffled and sorted by the framework. MapReduce Framework sorts the output based on **keys**. This sorted output becomes the input to the **reduce tasks**. Reduce task provides reduced output by combining the output of the various mappers. Job inputs and outputs are stored in a file system. MapReduce framework also takes care of the other tasks such as scheduling, monitoring, re-executing failed tasks, etc.

Hadoop Distributed File System and MapReduce Framework run on the same set of nodes. This configuration allows effective scheduling of tasks on the nodes where data is present **(Data Locality)**. This in turn results in very high throughput.

There are two daemons associated with MapReduce Programming. A single master **JobTracker** per cluster and one slave **TaskTracker** per cluster-node. The JobTracker is responsible for scheduling tasks to the TaskTrackers, monitoring the task, and re-executing the task just in case the TaskTracker fails. The TaskTracker executes the task. Refer Figure 5.21.

The MapReduce functions and input/output locations are implemented via the MapReduce applications. These applications use suitable interfaces to construct the job. The application and the job parameters together are known as **job configuration**. Hadoop **job client** submits job (jar/executable, etc.) to the JobTracker. Then it is the responsibility of JobTracker to schedule tasks to the slaves. In addition to scheduling, it also monitors the task and provides status information to the job-client.

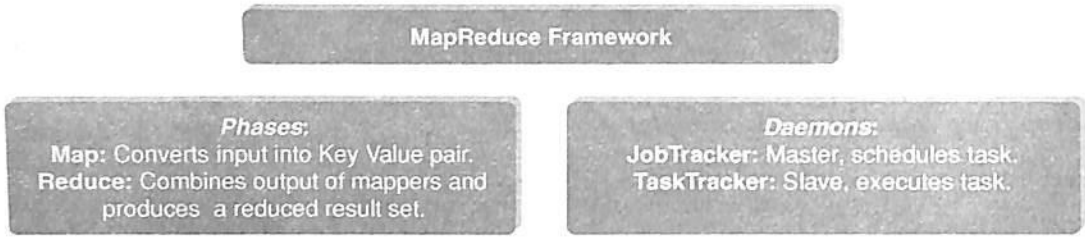


Figure 5.21 MapReduce Programming phases and daemons.

Reference: http://hadoop.apache.org/docs/r1.0.4/mapred_tutorial.html

5.11.1 MapReduce Daemons

1. **JobTracker:** It provides connectivity between Hadoop and your application. When you submit code to cluster, JobTracker creates the execution plan by deciding which task to assign to which node. It also monitors all the running tasks. When a task fails, it automatically re-schedules the task to a different node after a predefined number of retries. JobTracker is a master daemon responsible for executing overall MapReduce job. There is a single JobTracker per Hadoop cluster.
2. **TaskTracker:** This daemon is responsible for executing individual tasks that is assigned by the JobTracker. There is a single TaskTracker per slave and spawns multiple Java Virtual Machines (JVMs) to handle multiple map or reduce tasks in parallel. TaskTracker continuously sends heartbeat message to JobTracker. When the JobTracker fails to receive a heartbeat from a TaskTracker, the JobTracker assumes that the TaskTracker has failed and resubmits the task to another available node in the cluster. Once the client submits a job to the JobTracker, it partitions and assigns diverse MapReduce tasks for each TaskTracker in the cluster. Figure 5.22 depicts JobTracker and TaskTracker interaction.

Reference: Hadoop in Action, Chuck Lam.

5.11.2 How Does MapReduce Work?

MapReduce divides a data analysis task into two parts – **map** and **reduce**. Figure 5.23 depicts how the MapReduce Programming works. In this example, there are two mappers and one reducer. Each mapper

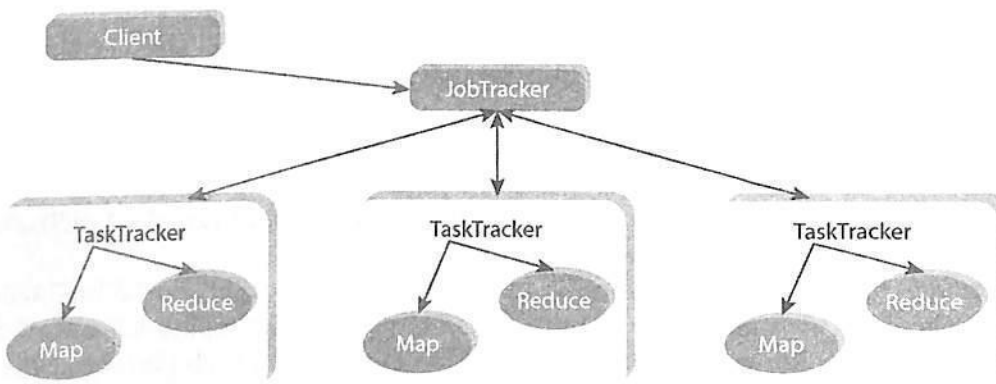


Figure 5.22 JobTracker and TaskTracker interaction.

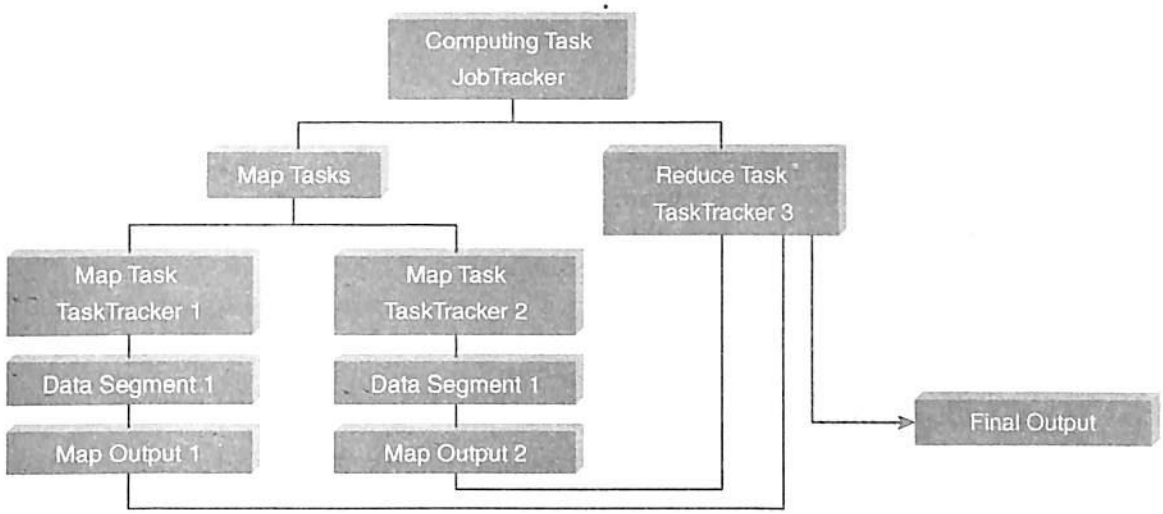


Figure 5.23 MapReduce programming workflow.

works on the partial dataset that is stored on that node and the reducer combines the output from the mappers to produce the reduced result set.

Reference: Wrox Big Data Certification Materials.

Figure 5.24 describes the working model of MapReduce Programming. The following steps describe how MapReduce performs its task.

1. First, the input dataset is split into multiple pieces of data (several small subsets).
2. Next, the framework creates a master and several workers processes and executes the worker processes remotely.

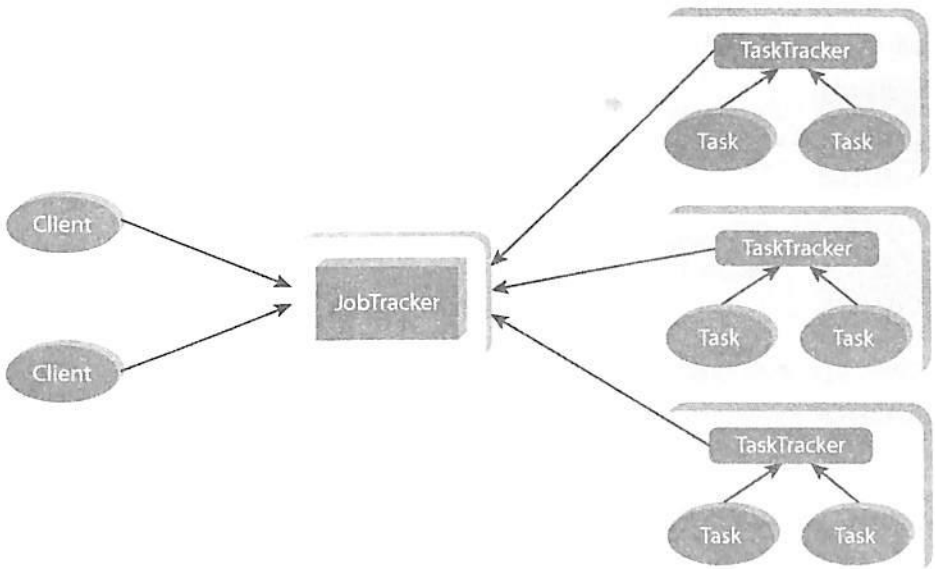


Figure 5.24 MapReduce programming architecture.

3. Several map tasks work simultaneously and read pieces of data that were assigned to each map task. The map worker uses the map function to extract only those data that are present on their server and generates key/value pair for the extracted data.
4. Map worker uses partitioner function to divide the data into regions. Partitioner decides which reducer should get the output of the specified mapper.
5. When the map workers complete their work, the master instructs the reduce workers to begin their work. The reduce workers in turn contact the map workers to get the key/value data for their partition. The data thus received is shuffled and sorted as per keys.
6. Then it calls reduce function for every unique key. This function writes the output to the file.
7. When all the reduce workers complete their work, the master transfers the control to the user program.

5.11.3 MapReduce Example

The famous example for MapReduce Programming is **Word Count.** For example, consider you need to count the occurrences of similar words across 50 files. You can achieve this using MapReduce Programming. Refer Figure 5.25.

Word Count MapReduce Programming using Java

The MapReduce Programming requires three things.

1. **Driver Class:** This class specifies **Job Configuration** details.
2. **Mapper Class:** This class overrides the **Map Function** based on the problem statement.
3. **Reducer Class:** This class overrides the **Reduce Function** based on the problem statement.

Wordcounter.java: Driver Program

```
package com.app;
import java.io.IOException;
```

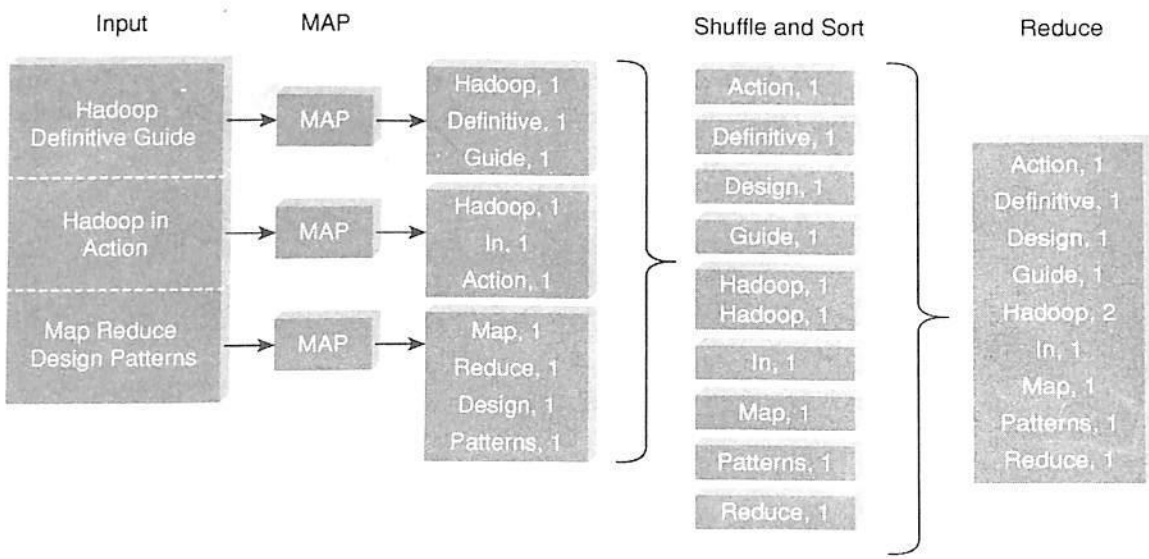


Figure 5.25 Wordcount example.

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCounter {

public static void main (String [] args) throws IOException,
InterruptedException, ClassNotFoundException {
          Job job = new Job ();
          job.setJobName ("wordcounter");
          job.setJarByClass (WordCounter.class);
          job.setMapperClass (WordCounterMap.class);
          job.setReducerClass (WordCounterRed.class);
          job.setOutputKeyClass (Text.class);
          job.setOutputValueClass (IntWritable.class);

          FileInputFormat.addInputPath (job, new Path ("/sample/word.
txt"));
          FileOutputFormat.setOutputPath (job, new Path ("/sample/
wordcount"));
          System.exit (job.waitForCompletion (true)? 0: 1);

     }
}
```

WordCounterMap.java: Map Class

```
package com.app;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCounterMap extends Mapper<LongWritable, Text, Text,
IntWritable> {
     @Override
```

```
    protected void map (LongWritable key, Text value, Context context)
              throws IOException, InterruptedException {

         String [] words=value.toString ().split (",");
         for (String word: words) {
              context.write (new Text (word), new IntWritable (1));
         }
    }
}
```

WordCountReduce.java: Reduce Class

```
package com.infosys;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCounterRed extends Reducer<Text, IntWritable, Text,
IntWritable>{

    @Override
    protected void reduce(Text word, Iterable<IntWritable> values,
Context context)
              throws IOException, InterruptedException {
         Integer count = 0;
         for(IntWritable val: values){
              count += val.get();
         }
         context.write(word, new IntWritable(count));
    }
}
```

Table 5.2 describes differences between SQL and MapReduce.

Table 5.2 SQL versus MapReduce

	SQL	MapReduce
Access	Interactive and Batch	Batch
Structure	Static	Dynamic
Updates	Read and write many times	Write once, read many times
Integrity	High	Low
Scalability	Nonlinear	Linear

5.12 MANAGING RESOURCES AND APPLICATIONS WITH HADOOP YARN (YET ANOTHER RESOURCE NEGOTIATOR)

Apache Hadoop YARN is a sub-project of Hadoop 2.x. Hadoop 2.x is YARN-based architecture. It is a general processing platform. YARN is not constrained to MapReduce only. You can run multiple applications in Hadoop 2.x in which all applications share a common resource management. Now Hadoop can be used for various types of processing such as Batch, Interactive, Online, Streaming, Graph, and others.

5.12.1 Limitations of Hadoop 1.0 Architecture

In Hadoop 1.0, HDFS and MapReduce are Core Components, while other components are built around the core.

1. Single NameNode is responsible for managing entire namespace for Hadoop Cluster.
2. It has a restricted processing model which is suitable for batch-oriented MapReduce jobs.
3. Hadoop MapReduce is not suitable for interactive analysis.
4. Hadoop 1.0 is not suitable for machine learning algorithms, graphs, and other memory intensive algorithms.
5. **MapReduce** is responsible for **cluster resource management and data processing.**

In this Architecture, **map slots might be "full", while the reduce slots are empty and vice versa.** This causes **resource utilization issues.** This needs to be improved for proper resource utilization.

5.12.2 HDFS Limitation

NameNode saves all its file metadata in main memory. Although the main memory today is not as small and as expensive as it used to be two decades ago, still there is a limit on the number of objects that one can have in the memory on a single NameNode. The NameNode can quickly become overwhelmed with load on the system increasing.

In Hadoop 2.x, this is resolved with the help of **HDFS Federation.**

5.12.3 Hadoop 2: HDFS

HDFS 2 consists of two major components: **(a) namespace, (b) blocks storage service.** Namespace service takes care of file-related operations, such as creating files, modifying files, and directories. The block storage service handles data node cluster management, replication.

HDFS 2 Features

1. Horizontal scalability.
2. High availability.

HDFS Federation uses multiple independent NameNodes for horizontal scalability. NameNodes are independent of each other. It means, NameNodes does not need any coordination with each other. The DataNodes are common storage for blocks and shared by all NameNodes. All DataNodes in the cluster registers with each NameNode in the cluster.

High availability of NameNode is obtained with the help of **Passive Standby NameNode.** In Hadoop 2.x, Active-Passive NameNode handles failover automatically. All namespace edits are recorded to a shared NFS storage and there is a single writer at any point of time. Passive NameNode reads edits from shared storage

and keeps updated metadata information. In case of Active NameNode failure, Passive NameNode becomes an Active NameNode automatically. Then it starts writing to the shared storage. Figure 5.26 describes the Active–Passive NameNode interaction.

Reference: http://www.edureka.co/blog/introduction-to-hadoop-2-0-and-advantages-of-hadoop-2-0/

Figure 5.27 depicts Hadoop 1.0 and Hadoop 2.0 architecture.

5.12.4 Hadoop 2 YARN: Taking Hadoop beyond Batch

YARN helps us to store all data in one place. We can interact in multiple ways to get predictable performance and quality of services. This was originally architected by **Yahoo**. Refer Figure 5.28.

Figure 5.26 Active and Passive NameNode interaction.

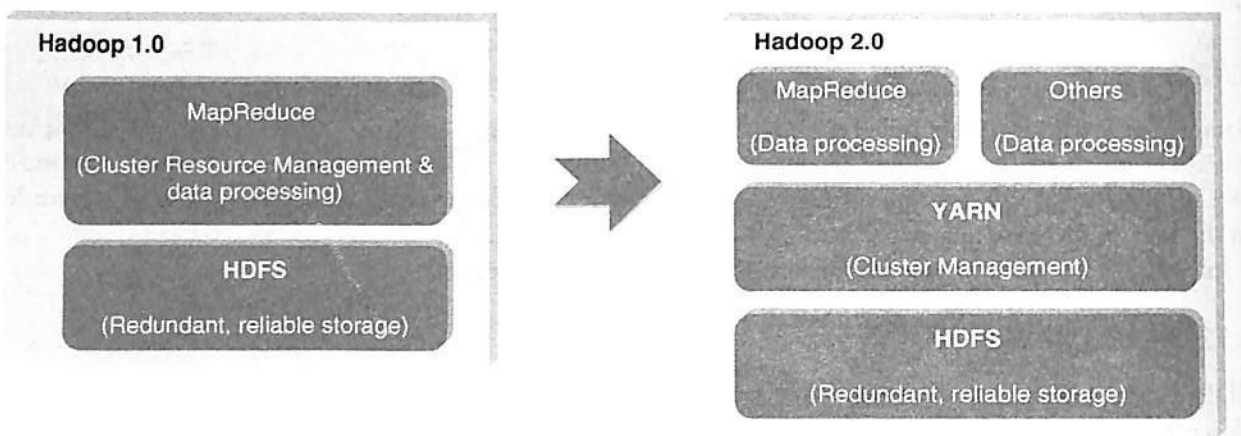


Figure 5.27 Hadoop 1.x versus Hadoop 2.x.

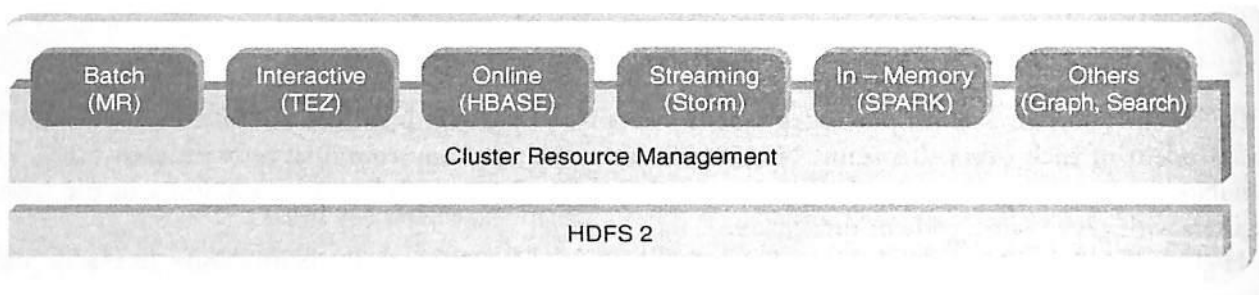


Figure 5.28 Hadoop YARN.

5.12.4.1 Fundamental Idea

The fundamental idea behind this architecture is splitting the JobTracker responsibility of resource management and Job Scheduling/Monitoring into separate daemons. Daemons that are part of YARN Architecture are described below.

1. **A Global ResourceManager:** Its main responsibility is to distribute resources among various applications in the system. It has two main components:
   (a) ***Scheduler:*** The pluggable scheduler of ResourceManager decides allocation of resources to various running applications. The scheduler is just that, a pure scheduler, meaning it does NOT monitor or track the status of the application.
   (b) ***ApplicationManager:*** ApplicationManager does the following:
   - Accepting job submissions.
   - Negotiating resources (container) for executing the application specific ApplicationMaster.
   - Restarting the ApplicationMaster in case of failure.
2. **NodeManager:** This is a per-machine slave daemon. NodeManager responsibility is launching the application containers for application execution. NodeManager monitors the resource usage such as memory, CPU, disk, network, etc. It then reports the usage of resources to the global ResourceManager.
3. **Per-application ApplicationMaster:** This is an application-specific entity. Its responsibility is to negotiate required resources for execution from the ResourceManager. It works along with the NodeManager for executing and monitoring component tasks.

5.12.4.2 Basic Concepts

Application:

1. Application is a job submitted to the framework.
2. Example – **MapReduce Job.**

Container:

1. Basic unit of allocation.
2. Fine-grained resource allocation across multiple resource types (Memory, CPU, disk, network, etc.)
   (a) container_0 = 2GB, 1CPU
   (b) container_1 = 1GB, 6 CPU
3. Replaces the fixed map/reduce slots.

YARN Architecture:

Figure 5.29 depicts YARN architecture. The steps involved in YARN architecture are as follows:

1. A client program submits the application which includes the necessary specifications to launch the application-specific **ApplicationMaster** itself.
2. The ResourceManager launches the ApplicationMaster by assigning some container.
3. The ApplicationMaster, on boot-up, registers with the ResourceManager. This helps the client program to query the ResourceManager directly for the details.
4. During the normal course, ApplicationMaster negotiates appropriate resource containers via the resource-request protocol.

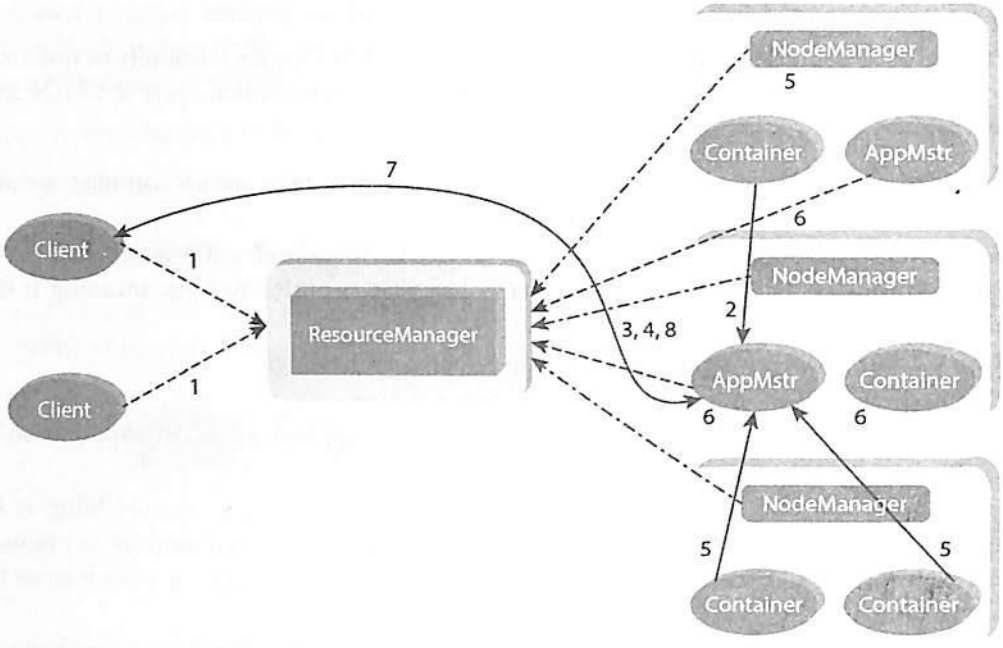


Figure 5.29 YARN architecture.

5. On successful container allocations, the ApplicationMaster launches the container by providing the container launch specification to the NodeManager.
6. The NodeManager executes the application code and provides necessary information such as progress, status, etc. to it's ApplicationMaster via an application-specific protocol.
7. During the application execution, the client that submitted the job directly communicates with the ApplicationMaster to get status, progress updates, etc. via an application-specific protocol.
8. Once the application has been processed completely, ApplicationMaster deregisters with the ResourceManager and shuts down, allowing its own container to be repurposed.

Reference: http://hortonworks.com/blog/apache-hadoop-yarn-background-and-an-overview/

5.13 INTERACTING WITH HADOOP ECOSYSTEM

Hadoop ecosystem was introduced in Chapter 4. Here we will look at it in more detail.

5.13.1 Pig

Pig is a data flow system for Hadoop. It uses Pig Latin to specify data flow. Pig is an alternative to MapReduce Programming. It abstracts some details and allows you to focus on data processing. It consists of two components.

1. **Pig Latin:** The data processing language.
2. **Compiler:** To translate Pig Latin to MapReduce Programming.

Figure 5.30 depicts the Pig in the Hadoop ecosystem.

5.13.2 Hive

Hive is a Data Warehousing Layer on top of Hadoop. Analysis and queries can be done using an SQL-like language. Hive can be used to do ad-hoc queries, summarization, and data analysis. Figure 5.31 depicts Hive in the Hadoop ecosystem.

5.13.3 Sqoop

Sqoop is a tool which helps to transfer data between Hadoop and Relational Databases. With the help of Sqoop, you can import data from RDBMS to HDFS and vice-versa. Figure 5.32 depicts the Sqoop in Hadoop ecosystem.

5.13.4 HBase

HBase is a NoSQL database for Hadoop. HBase is column-oriented NoSQL database. HBase is used to store **billions of rows and millions of columns.** HBase provides random read/write operation. It also supports record level updates which is not possible using HDFS. HBase sits on top of HDFS. Figure 5.33 depicts the HBase in Hadoop ecosystem.

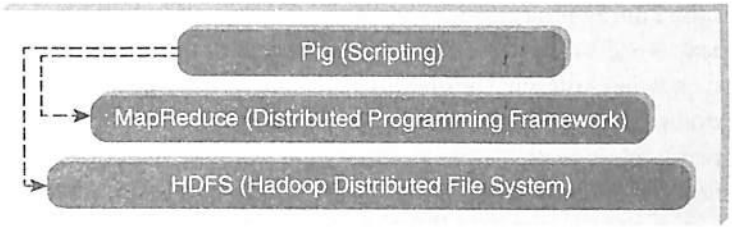


Figure 5.30 Pig in the Hadoop ecosystem.

Figure 5.31 Hive in the Hadoop ecosystem.

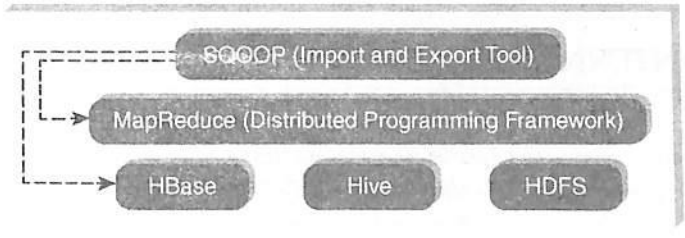


Figure 5.32 Sqoop in the Hadoop ecosystem.

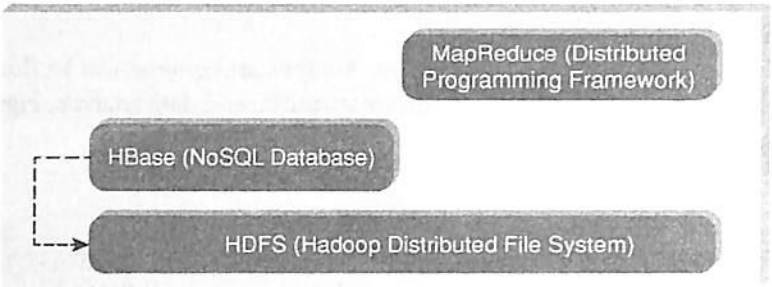


Figure 5.33 HBase in the Hadoop Ecosystem.

REMIND ME

- The key consideration (the rationale behind the huge popularity of Hadoop) is: ***Its capability to handle massive amounts of data, different categories of data – fairly quickly.***
- Hadoop was created by Doug Cutting, the creator of Apache Lucene (a commonly used text search library). Hadoop is a part of the Apache Nutch (Yahoo) project (an open-source web search engine) and also a part of the Lucene project.
- Hadoop is an open-source software framework. It stores and processes huge volumes of data in a distributed fashion on large clusters of commodity hardware. Basically, Hadoop accomplishes two tasks:
  - Massive data storage.
  - Faster data processing.
- The core components of Hadoop are:
  - HDFS
  - MapReduce
- Apache Hadoop YARN is a sub-project of Hadoop 2.x. Hadoop 2.x is YARN-based architecture. It provides general processing platform which is not constrained to MapReduce only.

POINT ME (BOOKS)

- Hadoop, the Definite Guide, 3rd Edition, O'reilly Publication.
- Hadoop in Practice, Alex Holmes.
- Hadoop in Action, Chuck Lam.

CONNECT ME (INTERNET RESOURCES)

- http://hadoop.apache.org/docs/r1.0.4/hdfs_design.html
- http://hadoop.apache.org/docs/r1.0.4/mapred_tutorial.html
- http://oraclesys.com/2013/04/03/difference-between-hadoop-and-rdbms/

- http://hortonworks.com/blog/apache-hadoop-yarn-background-and-an-overview/
- http://www.tomsitpro.com/articles/hadoop-2-vs-1,2-718.html
- http://www.wikidifference.com/difference-between-hadoop-and-rdbms/
- http://www.edureka.co/blog/introduction-to-hadoop-2-0-and-advantages-of-hadoop-2-0/

TEST ME

A. Fill Me

1. Hadoop is _____ based flat structure.
2. RDBMS is best choice when _____ is the main concern.
3. Hadoop supports _____, _____ and _____ data formats.
4. RDBMS supports _____ data formats.
5. In Hadoop, data is processed in _____.
6. HDFS can be deployed on _____.
7. NameNode uses _____ to store file system namespace.
8. NameNode uses _____ to record every transaction.
9. Secondary NameNode is a _____ daemon.
10. DataNode is responsible for _____ file operation.
11. Hadoop 2.x is based on _____ architecture.
12. YARN is responsible for _____.
13. Global ResourceManager distributes _____ among applications.
14. NodeManager is responsible for launching Application _____.
15. Application is a _____ submitted to framework.
16. _____ is an open-source framework managed by Apache Software Foundations.
17. The emphasis of HDFS is on _____ throughput of data access rather than _____ latency of data access.
18. An HDFS cluster consists of a single _____ and a number of _____.
19. Complete the series:
    Bits → Bytes → Kilobytes → Megabytes → Gigabytes → _____ → _____ → _____ → _____ → Yottabytes
20. HDFS has a _____/_____ architecture.
21. HDFS is built using the _____ language.
22. The _____ maintains the file system Namespace.
23. The number of copies of a file is called the _____ of that file.
24. The NameNode periodically receives a _____ and a _____ from each of the DataNodes in the cluster.
25. Receipt of a Heartbeat implies that the _____ is functioning properly.
26. A _____ contains a list of all blocks on a DataNode.
27. The blocks of a file are replicated for _____ tolerance.
28. When the NameNode starts up, it reads the _____ and _____ from disk.
29. A typical block size used by HDFS is _____.
30. _____ are responsible for serving read and write requests from the file system's clients.

31. _____ perform block creation, deletion and replication upon instruction from the _____.
32. _____ was the first to publicize MapReduce – a system they had used to scale their data processing needs.
33. _____ developed an open-source version of MapReduce system called _____.
34. Hadoop is an open-source framework for writing and running _____ applications that process large amounts of data.
35. The key distinctions of Hadoop are _____, _____ and _____.
36. Hadoop runs on large clusters of _____.
37. Hadoop scales _____ to handle larger data by adding more _____ to the cluster.
38. Hadoop focusses on moving _____ to _____.
39. The move-code-to-data philosophy makes sense for _____ intensive processing.
40. Hadoop is designed to be a scale _____ architecture operating on cluster of commodity PC machines.
41. Hadoop uses _____ as its basic data unit, which is flexible enough to work with less-structured data types.
42. Hadoop is best used as a _____ once and _____ many times type of data store.
43. Under SQL we have _____ statements; under MapReduce we have _____ and _____.
44. Under the MapReduce Model, data processing primitives are called _____ and _____.
45. The Mapper is meant to _____ and _____ the input into something that the reducer can _____ over.
46. _____ and _____ are common design patterns that go along with mapping and reducing.
47. _____ is the official development and production platform for Hadoop.
48. _____ started out as a sub-project of _____, which in turn was a sub-project of _____.
49. _____ is a single point of failure of Hadoop cluster.
50. _____ is the bookkeeper of HDFS.
51. _____ keeps track of how your files are broken down into file blocks, which nodes store those blocks, and the overall health of the distributed file system.
52. _____ communicates with the NameNode to take snapshots of the HDFS metadata at intervals defined by the cluster configuration.
53. There is only one _____ daemon per Hadoop cluster.
54. There is a single _____ per slave node.

Answers:

1. Node
2. Consistency
3. Structured, semi-structured and unstructured
4. Structured
5. Parallel
6. Low cost hardware
7. FsImage
8. EditLog

9. Helper or House Keeping
10. Read/Write
11. YARN
12. Cluster Management
13. Resources
14. Containers
15. Job
16. Hadoop
17. High, Low
18. NameNode, DataNodes
19. Terabytes, Petabytes, Exabytes, Zettabytes
20. Master/slave
21. Java
22. NameNode
23. Replication factor
24. Heartbeat, Blockreport
25. DataNode
26. Blockreport
27. Fault
28. FsImage, EditLog
29. 64MB
30. DataNodes
31. DataNodes, NameNode
32. Google
33. Doug Cutting, Hadoop
34. Distributed
35. Accessible, Robust, and Scalable
36. Commodity machines
37. Linearly, nodes
38. Code, Data
39. Data
40. Out
41. Key/value
42. Write, read
43. Query, Scripts, and Code
44. Mappers, Reducers
45. Filter and transform, aggregate
46. Partitioning and Shuffling
47. Linux
48. Hadoop, Nutch, Apache Lucene
49. NameNode
50. NameNode
51. NameNode
52. Secondary NameNode
53. JobTracker
54. TaskTracker

B. Match Me

1.

Column A	Column B
HDFS	DataNode
MapReduce Programming	NameNode
Master node	Processing Data
Slave node	Google File System and MapReduce
Hadoop Implementation	Storage

Answer:

Column A	Column B
HDFS	Storage
MapReduce Programming	Processing Data
Master node	NameNode
Slave node	DataNode
Hadoop Implementation	Google File System and MapReduce

2.

Column A	Column B
JobTracker	Executes Task
MapReduce	Schedules Task
TaskTracker	Programming Model
Job Configuration	Converts input into Key Value pair
Map	Job Parameters

Answer:

Column A	Column B
JobTracker	Schedules Task
MapReduce	Programming Model
TaskTracker	Executes Task
Job Configuration	Job Parameters
Map	Converts input into Key Value pair

3.

Column A	ColumnB
NameNode	Handles processing on master
JobTracker	Handles storage on slave
DataNode	Handles storage on master
TaskTracker	Handles processing on slave

Answer:

Column A	Column B
NameNode	Handles storage on master
JobTracker	Handles processing on master
DataNode	Handles storage on slave
TaskTracker	Handles processing on slave

C. True or False

1. For using Hadoop to process your data, the data has to be moved/ingested into HDFS.
2. Sqoop is used to query HDFS data.

3. Oozie is to import/export data from RDBMS.
4. "hadoop fs -ls /" will show the contents for the HDFS root directory.
5. Master node in Hadoop can be low on disk space but needs to have good amount of RAM.
6. In Production, NameNode preferably runs on Red Hat OS.
7. Hadoop configurations are stored in CSV format.

Answers:

1. True
2. False
3. False
4. True
5. True
6. True
7. False

D. Pick the Right Choice

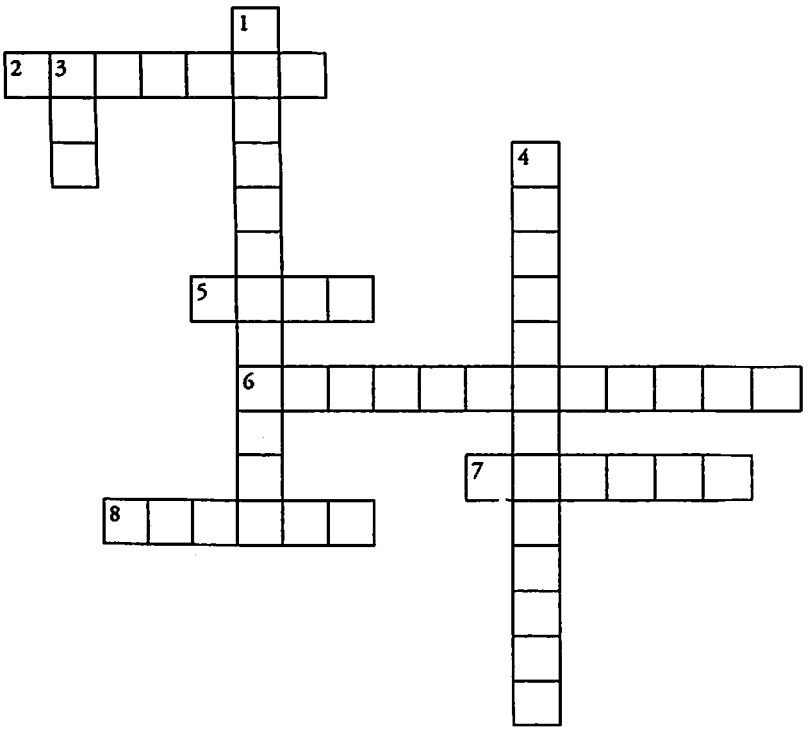
1. Which of the two are components of Hadoop?
   (a) HDFS
   (b) MapReduce
   (c) Secondary NameNode
   (d) Shuffler
   (e) Sqoop
2. How many blocks will be created for a file that is 300 MB? The default block size is 64 MB and the replication factor is 3.
   (a) 30
   (b) 5
   (c) 15
   (d) 100
3. Pig is a
   (a) Data flow language
   (b) Scheduling engine
   (c) Import export tool
   (d) Shuffler
4. What does JobTracker do?
   (a) Stores blocks of data
   (b) Coordinates and schedules the job
   (c) Stores metadata
   (d) Acts as a mini reducer
5. Which ecosystem project is ideal for use when we have multiple MapReduce and Pig programs to run in a sequence?
   (a) Oozie
   (b) Pig
   (c) Hive
   (d) Sqoop
6. Which file is used for updating MapReduce settings?
   (a) core-site
   (b) hdfs-site
   (c) mapred-site
   (d) hadoop-env.sh

Answers:

1. (a) and (b)
2. (c)
3. (a)
4. (b)
5. (a)
6. (c)

E. Crossword

Puzzle on Big Data and Hadoop
Complete the crossword below



Across

2. One _____ Gigabytes are there in one Exabyte.
5. _____ is Splunk's new product to search, access and report on Hadoop data sets.
6. _____ gave Hadoop its name
7. _____ open-source software was developed from Google's MapReduce concept.
8. The MapReduce programming model widely used in analytics was developed at _____.

Down

1. _____ created the popular Hadoop software framework for storage and processing of large datasets.
3. _____ traditional IT Company is the largest Big Data vendor in the world.
4. According to a study by IBM, approximately _____ amount of data existed in the digital universe in 2012.

Answer:

Across

2. Billion
5. Hunk
6. Toy Elephant
7. Hadoop
8. Google

Down

1. Doug cutting
3. IBM
4. 2.7 Zettabytes

CHALLENGE ME

There are questions on topics that are not covered in the chapter. We will need you to read up on your own.

1. **What are the four modules that make up the Apache Hadoop framework?**

   Answer:

   - Hadoop Common, which contains the common utilities and libraries necessary for Hadoop's other modules.
   - Hadoop YARN, the framework's platform for resource management.
   - Hadoop Distributed File System, or HDFS, which stores information on commodity machines.
   - Hadoop MapReduce, a programming model used to process large-scale sets of data.

2. **Which modes can Hadoop be run in? List a few features for each mode.**

   Answer:

   - Standalone, or local mode, which is one of the least commonly used environments. When it is used, it's usually only for running MapReduce programs. Standalone mode lacks a distributed file system and uses a local file system instead.
   - Pseudo-distributed mode, which runs all daemons on a single machine. It is most commonly used in QA and development environments.
   - Fully distributed mode, which is most commonly used in production environments. Unlike pseudo-distributed mode, fully distributed mode runs all daemons on a cluster of machines rather than a single one.

3. **Where are Hadoop's configuration files located?**

   Answer: Hadoop's configuration files can be found inside the conf sub-directory.

4. **List Hadoop's three configuration files.**

   Answer:

   - hdfs-site.xml
   - core-site.xml
   - mapred-site.xml

5. **How many NameNodes can run on a single Hadoop cluster?**

   Answer: Only one NameNode process can run on a single Hadoop cluster. The file system will go offline if this NameNode goes down.

6. **What is a DataNode?**

   Answer: Unlike NameNode, a DataNode actually stores data within the Hadoop distributed file system. DataNodes run on their own Java virtual machine process.

7. **How many data nodes can run on a single Hadoop cluster?**

   Answer: Hadoop slave nodes contain only one data node process each.

8. **What is JobTracker in Hadoop?**

   Answer: JobTracker is used to submit and track jobs in MapReduce.

9. **How many JobTracker processes can run on a single Hadoop cluster?**

   Answer: There can only be one JobTracker process running on a single Hadoop cluster. JobTracker processes run on their own Java virtual machine process. If JobTracker goes down, all currently active jobs stop.

10. What is the difference between replication and sharding?

Answer: Replication essentially takes the same data and copies it over several machines/nodes (the number of copies it makes, depends on the defined replication factor).

Sharding takes different data and places it on different machines. It is particularly valuable for performance as it can help with read and write operations. Replication is for fault tolerance.

11. What is polyglot persistence?

Answer: The official definition of polyglot is "a person who has the ability to speak, read, and write several languages". Now consider an organization that has grown over 35 years. It has a lot of applications which write to a number of data sources (RDBMSs, Flat files, .xls, csv files, etc.). The organization also has several data marts, content management server, etc. This is a typical polyglot situation as an analytics application may require the data to be read from all of these different types of data sources.

Consider a scenario: You are one of the sponsors of an online retail firm.
You have a few questions which you need answered:

- Who are the customers who have purchased a product X in the last 12 months?
- Do you have comments left by these customers on social network site?
- Are there repeat customers on the company's website?
- Have they recommended your product to their friends, colleagues, and relatives?
- Did they go to check the product elsewhere?

This calls for data to be collected from varied disparate data sources (relational and non-relational) and analyzed. The above is a typical case of polyglot persistence.

12. What is BigTable?

Answer: It is a compressed, proprietary data storage system built on Google File System. It is not distributed outside of Google, although it underlies the Google Datastore.