VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY
(Autonomous)

Department of Computer Science and Engineering

IV B.Tech – I Semester (Sections – C & D)

Big Data Analytics (Elective – I)

UNIT – II : Working with Big Data

Syllabus

UNIT-II

Working with Big Data:

Google File System, Hadoop Distributed File System (HDFS), Building blocks of Hadoop (Namenode, Datanode, Secondary Namenode, JobTracker, TaskTracker), Introducing and Configuring Hadoop cluster (Local, Pseudo-distributed mode, Fully Distributed mode), Configuring XML files.

Outline of Unit-II

- Popular Software Tools for Data Analytics
- Introduction to Big Data Analytics
- The 8 V's of Big Data
- Google File System (GFS)
- Hadoop Introduction & Overview
- Apache Hadoop Ecosystem
- Hadoop Distributed File System (HDFS)
- Building blocks of Hadoop

  Namenode, Datanode, Secondary Namenode, JobTracker, TaskTracker
- Introducing and Configuring Hadoop cluster

  Local, Pseudo-distributed mode, Fully Distributed mode
- Configuring XML files

Popular Software Tools for Data Analytics

https://www.softwaretestinghelp.com/big-data-tools/

Introduction to Big Data Analytics

https://www.guru99.com/what-is-big-data.html

The 8 V’s of Big Data

The 8 V’s are:

1. **Volume**
2. **Velocity**
3. **Variety**
4. Veracity
5. Vocabulary
6. Vagueness
7. Viability
8. Value

Volume

The amount of data needing to be processed at a given time. This can marked either as amount over time or amount that needs to be processed at one time.

For example, doing a matrix operation on a 1 billion by 1 billion matrix (or) scanning the contents of every published newspaper in a day for key words are both examples of volume that can constrain computing.

Velocity

Similar to Volume, this has to do with the speed of the data coming in and the speed of the transformed data leaving the compute.

An example of a high velocity requirement is telemetry that needs to be analyzed in real time for a self-driving car. The enemy of velocity is latency.

Variety

The spice of life, or the bane of computing? In the computing context we are discussing, this term refers to heterogeneous data sources that need to be identified and normalized before the compute can occur.

In data science, this is often referred to as data cleaning, this operation is frequently the most labor intensive as it involves all of the pre-work required to set-up the high-performance compute.

This is where the vast majority of errors and issues are found with data and this is the fundamental bottle neck in high-performance computing.

Veracity

Big Data Veracity refers to the biases, noise and abnormality in data.

Is the data that is being stored, and mined meaningful to the problem being analyzed. The veracity in data analysis is the biggest challenge when compares to things like volume and velocity.

In scoping out your big data strategy you need to have your team and partners work to help keep your data clean and processes to keep ‘dirty data’ from accumulating in your systems.

Vocabulary

This term has two meanings. The first meaning is less a computing issue than it is a communication issue between provider and customer and it has to do with the language used to describe the desired outcome of an analysis.

For example, the term "accuracy" or "performance" may have different meaning in the context of structural engineering than it does in rendering animation.

The second meaning branches into semantic searching and operations within a semantic space.

Vocabulary – Contd.

Here we are dealing with controlled vocabularies (ontologies) that represent a specific definition but also a relatedness to another term.

For example, the term “child” infers that it has a “parent” and so forth. This term architecture is very important when operating with clients in the artificial intelligence space where search and retrieval is used to uncover unknown relationships.

As it turns out, the strength of the ontology is what leads to the relative success or failure in projects that mine with semantic-based technologies.

Vagueness

This term describes an interpretation issue with results being returned.

This is a bit tongue-in-cheek, but, it is a very real problem with scientific and big data computes.

Viability

This refers to a model's ability to represent reality. Model's by their very nature are idealized approximations of reality.

Some are very good, others are all dangerously flawed. Frequently, model builders simplify their models in order for them to be computationally tractable.

With hardware acceleration, we can remove these shackles from the model builder and let them simulate closer to reality.

Value

This term is defined as whatever is important to the customer.

Another way to define value is the removal of obstacles in their path to allow them to get to their stated destination.

We often think of value in terms of cost, but, we can also think of Value in terms of enablement and what that is worth to the customer.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

The V’s of Big Data

https://www.kdnuggets.com/2017/04/42-vs-big-data-data-science.html

Introduction to BIG DATA

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

- Big Data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn't fit the strictures [restrictions] of our existing database architectures. To gain value from this data, you must choose an alternative way to process it.
- The hot IT buzzword, Big Data has become viable as cost-effective approaches have emerged to tame the Volume, Velocity and Variability of massive data. Within this data lie valuable patterns and information, previously hidden because of the amount of work required to extract them. To leading corporations, like Walmart or Google, this power has been in reach for some time, but at fantastic cost.

Introduction to BIG DATA – Contd.

- Today's commodity hardware, cloud architectures and open source software bring big data processing into the reach of the less well-resourced. Big data processing is eminently feasible for even the small garage startups, who can cheaply rent server time in the cloud.
- The value of big data to an organization falls into two categories:

  i) analytical use and

  ii) enabling new products.
- Big data analytics can reveal insights hidden previously by data too costly to process, such as peer influence among customers, revealed by analyzing shopper's transactions, social and geographical data.

Introduction to BIG DATA – Contd.

- Being able to process every item of data in reasonable time removes the troublesome need for sampling and promotes an investigative approach to data, in contrast to the somewhat static nature of running predetermined reports.
- The past decades successful web startups are prime examples of big data used as an enabler of new products and services. For example, by combining a large number of signals from a user's actions and those of their friends, Facebook has been able to craft a highly personalized user experience and create a new kind of advertising business. It's no coincidence that the lion's share of ideas and tools underpinning big data have emerged from Google, Yahoo, Amazon and Facebook.

Google File System (GFS)

- Google File System (GFS or Google FS) is a proprietary distribute file system developed by Google for its own usage purpose. It is designed to provide efficient, reliable access to data using large clusters of commodity hardware.
- GFS is enhanced for Google's core data storage and usage needs (primarily the search engine), which can generate enormous amounts of data that needs to be retained. Google File System grew out of an earlier Google effort, "BigFiles", developed by Larry Page and Sergey Brin in the early days of Google, while it was still located in Stanford. Files are divided into fixed size chunks of 64 megabytes, similar to clusters or sectors in regular file systems, which are only extremely rarely overwritten, or shrunk (become smaller); files are usually appended to or read. It is also designed and optimized to run on Google's computing clusters, dense nodes which consist of cheap "commodity" computers, which means precautions must be taken against the high failure rate of individual nodes and the subsequent data loss. Other design decisions select for high data throughputs, even when it comes at the cost of latency.

VVIT
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Google File System (GFS) – Contd.

- A GFS cluster consists of multiple nodes. These nodes are divided into 2 types: i) Master Node and ii) A large number of Chunkservers.

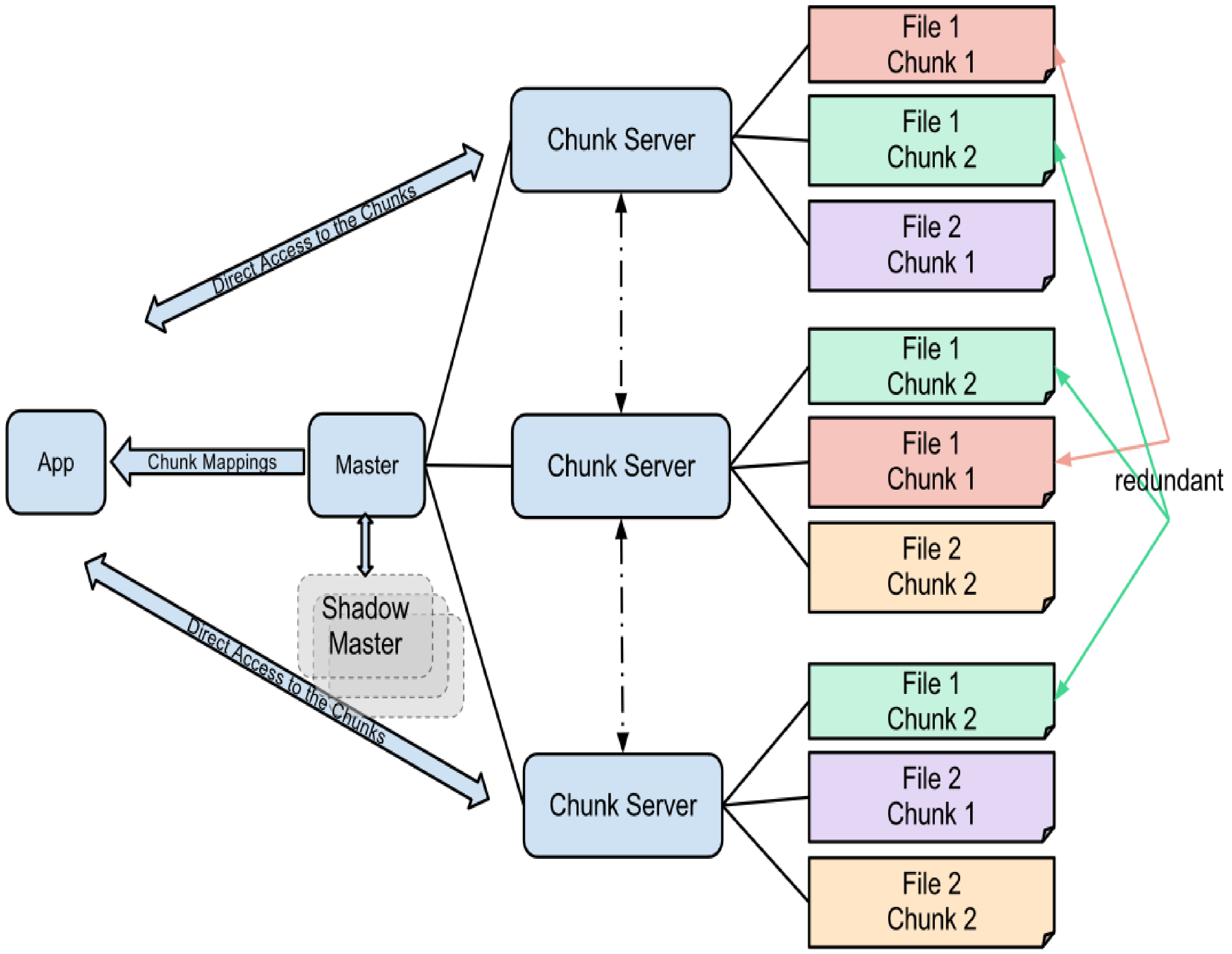
- Each file is divided into fixed size chunks. Chunk servers store these chunks. Each chunk is assigned a unique 64bit label by the master node at the time of creation, and logical mappings of files to constituent chunks are maintained. Each chunk is replicated several times throughout the network, with the minimum being three, but even more for files that have high end-in demand or need more redundancy. The Master server does not usually store the actual chunks, but rather all the metadata associated with the chunks, such as the tables mapping the 64bit labels to chunk locations and the files they make up, the locations of the copies of the chunks, what processes are reading or writing to a particular chunk, or taking a "snapshot" of the chunk pursuant to replicate it (usually at the instigation of the Master server, when, due to node failures, the number of copies of a chunk has fallen beneath the set number).

Google File System (GFS) – Contd.

- All this metadata is kept current by the Master server periodically receiving updates from each chunk server ("Heartbeat messages"). Permissions for modifications are handled by a system of time-limited, expiring "leases", where the Master server grants permission to a process for a finite period of time during which no other process will be granted permission by the Master server to modify the chunk. The modifying chunkserver, which is always the primary chunk holder, then propagates the changes to the chunkservers with the backup copies. The changes are not saved until all chunkservers acknowledge, thus guaranteeing the completion and atomicity of the operation. Programs access the chunks by first querying the Master server for the locations of the desired chunks; if the chunks are not being operated on (i.e. no outstanding leases exist), the Master replies with the locations, and the program then contacts and receives the data from the chunkserver directly (similar to Kazaa and its supernodes). Unlike most other file systems, GFS is not implemented in the kernel of an operating system, but is instead provided as a userspace library.

- The below Figure Shows the GFS architecture.

Google File System (GFS) – Contd.



HADOOP – Introduction & Overview

- Big Data is a collection of datasets so large and complex that it becomes difficult to store and process using on-hand database management tools or traditional data processing applications. Big Data Analytics consists the following:
  - Examining large amount of data.
  - Appropriate information.
  - Identification of hidden patterns, unknown correlations.
  - Competitive advantage.
  - Better business decisions: strategic and operational.
  - Effective marketing, customer satisfaction, increased revenue.
- To do analysis with big data, we need some tools which are available in real time. Out of them, Hadoop was became as popular, important and effective tools for big data analytics.

Brief History of HADOOP

- Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library. Hadoop has its origins in Apache Nutch, an open source web search engine, itself a part of the Lucene project.

The Origin of the Name “Hadoop”:

- The name Hadoop is not an acronym; it’s a made-up name. The project’s creator, Doug Cutting, explains how the name came about:
- The name his 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. 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). Smaller components are given more descriptive (and therefore more mundane) names. This is a good principle, as it means you can generally work out what something does from its name. For example, the jobtracker keeps track of MapReduce jobs.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Brief History of HADOOP – Contd.

2020	→	Hadoop still stands as one of the Top Analytics Tool.
2013	→	Hadoop used by Hundreds of Companies.
2009	→	Yahoo used Hadoop to sort 1 TB in 62 Secs.
2008	→	Hadoop become Apache Top Level Project.
2006	→	Yahoo hires Doug Cutting to work on Hadoop with a dedicated team.
2005	→	Doug Cutting & Nutch team implemented Google's frameworks in Nutch.
2004	→	Google publishes Google File System (GFS) & Map Reduce Framework Papers.

The core Hadoop contains Two components:

i) HDFS (Hadoop Distributed File System)

ii) MapReduce (Distributed Data Processing Model)

Brief History of HADOOP – Contd.

Advantages of Hadoop:

Some of the advantages of Hadoop was listed below:

Accessible

- Available
- Scalable
- Reliable
- Robust
- Simple
- Cost Effective
- Flexible
- Fast
- Resilient to Failures

Brief History of HADOOP – Contd.

- Before the year 2000, data was relatively small than it is currently; however, data computation was complex. All data computation was dependent on the processing power of the available computers.
- Later as data grew, the solution was to have computers with large memory and fast processors. However, after 2000, data kept growing and the initial solution could no longer help.
- Over the last few years, there has been an incredible explosion in the volume of data. IBM reported that 2.5 exabytes, or 2.5 billion gigabytes, of data, was generated every day in 2012.
- Here are some statistics indicating the proliferation of data from Forbes, September 2015. 40,000 search queries are performed on Google every second. Up to 300 hours of video are uploaded to YouTube every minute.

Brief History of HADOOP – Contd.

- In Facebook, 31.25 million messages are sent by the users and 2.77 million videos are viewed every minute. By 2017, nearly 80% of photos will be taken on smartphones.
- By 2020, at least a third of all data will pass through the Cloud (a network of servers connected over the Internet). By the year 2020, about 1.7 megabytes of new information will be created every second for every human being on the planet.
- Data is growing faster than ever before. You can use more computers to manage this ever-growing data. Instead of one machine performing the job, you can use multiple machines. This is called a distributed system.

Brief History of HADOOP – Contd.

How Does a Distributed System Work?

- Suppose you have one machine which has four input/output channels. The speed of each channel is 100 MB/sec and you want to process one terabyte of data on it.
- It will take 45 minutes for one machine to process one terabyte of data. Now, let us assume one terabyte of data is processed by 100 machines with the same configuration.
- It will take only 45 seconds for 100 machines to process one terabyte of data. Distributed systems take less time to process Big Data.
- Now, let us look at the challenges of a distributed system.

Brief History of HADOOP – Contd.

Challenges of Distributed Systems

- Since multiple computers are used in a distributed system, there are high chances of system failure. There is also a limit on the bandwidth.

- Programming complexity is also high because it is difficult to synchronize data and process. Hadoop can tackle these challenges.

Brief History of HADOOP – Contd.

What is Hadoop?

- Hadoop is a framework that allows for the distributed processing of large datasets across clusters of computers using simple programming models. It is inspired by a technical document published by Google.
- The word Hadoop does not have any meaning. Doug Cutting, who discovered Hadoop, named it after his son yellow-colored toy elephant.
- Let us discuss how Hadoop resolves the three challenges of the distributed system, such as high chances of system failure, the limit on bandwidth, and programming complexity.

Brief History of HADOOP – Contd.

The major four key characteristics of Hadoop are:

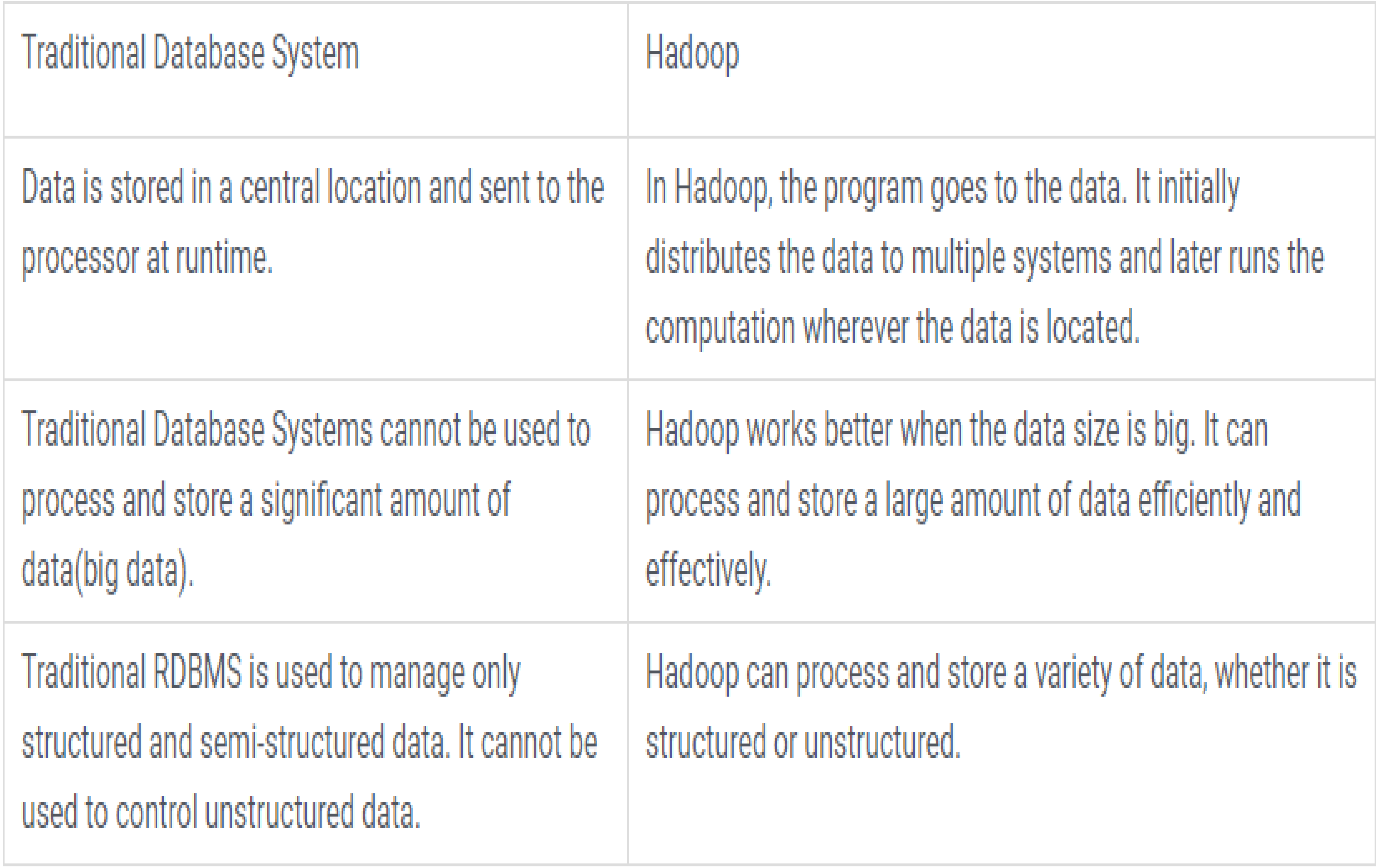
- Economical: Its systems are highly economical as ordinary computers can be used for data processing.
- Reliable: It is reliable as it stores copies of the data on different machines and is resistant to hardware failure.
- Scalable: It is easily scalable both, horizontally and vertically. A few extra nodes help in scaling up the framework.
- Flexible: It is flexible and you can store as much structured and unstructured data as you need to and decide to use them later.

Brief History of HADOOP – Contd.

- Traditionally, data was stored in a central location, and it was sent to the processor at runtime. This method worked well for limited data.
- However, modern systems receive terabytes of data per day, and it is difficult for the traditional computers or Relational Database Management System (RDBMS) to push high volumes of data to the processor.
- Hadoop brought a radical approach. In Hadoop, the program goes to the data, not vice versa. It initially distributes the data to multiple systems and later runs the computation wherever the data is located.
- Next, we will talk about how Hadoop differs from the traditional Database System.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Brief History of HADOOP – Contd.



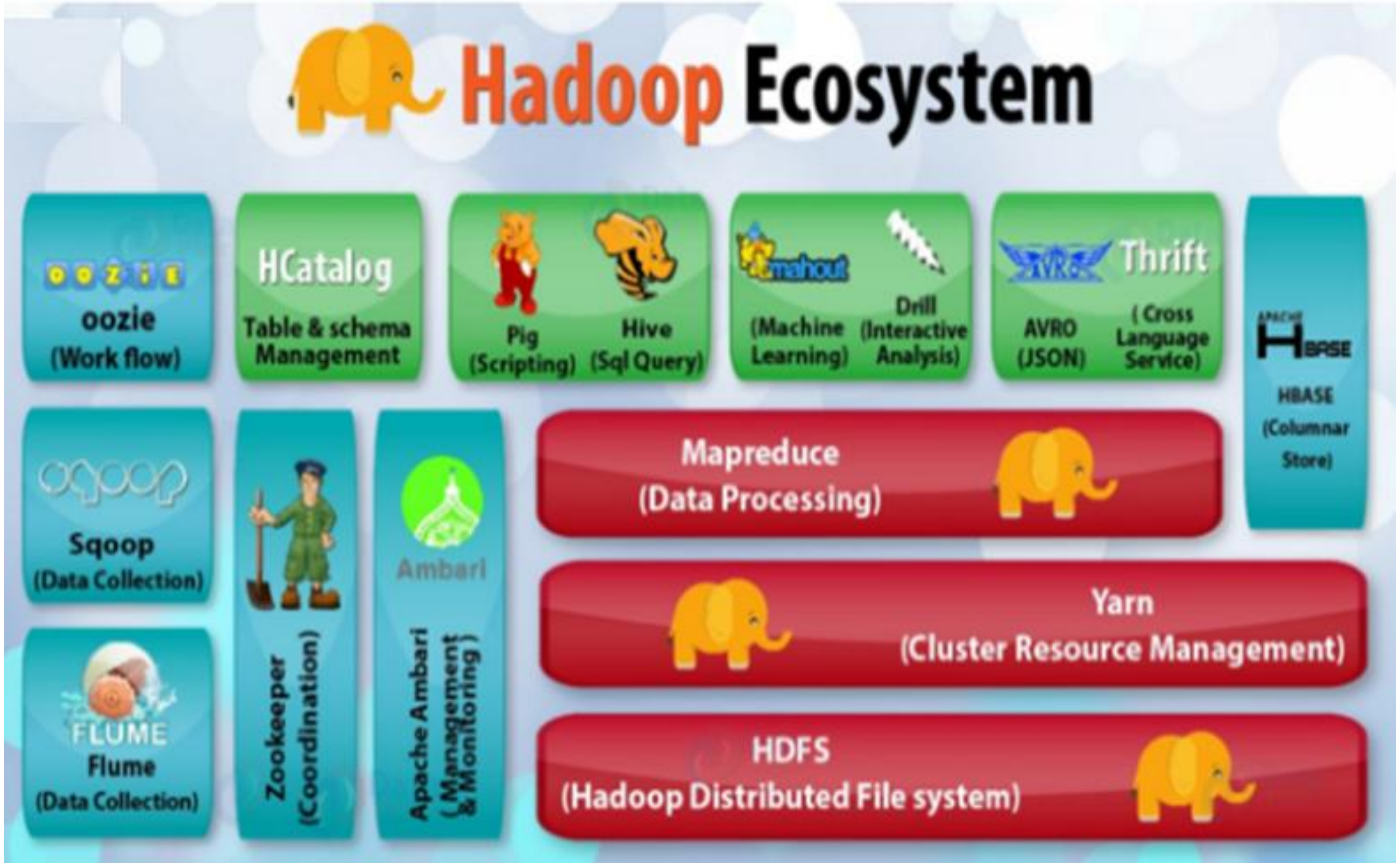
Traditional Database System	Hadoop
Data is stored in a central location and sent to the processor at runtime.	In Hadoop, the program goes to the data. It initially distributes the data to multiple systems and later runs the computation wherever the data is located.
Traditional Database Systems cannot be used to process and store a significant amount of data(big data).	Hadoop works better when the data size is big. It can process and store a large amount of data efficiently and effectively.
Traditional RDBMS is used to manage only structured and semi-structured data. It cannot be used to control unstructured data.	Hadoop can process and store a variety of data, whether it is structured or unstructured.

Apache Hadoop Ecosystem

- Hadoop Ecosystem Hadoop has an ecosystem that has evolved from its three core components processing, resource management, and storage. In this topic, you will learn the components of the Hadoop ecosystem and how they perform their roles during Big Data processing.

- The Hadoop ecosystem is continuously growing to meet the needs of Big Data. It comprises the following components:

Apache Hadoop Ecosystem – Contd.



Hadoop Ecosystem and Their Components

Apache Hadoop Ecosystem – Contd.

The Hadoop Ecosystem consists of several Hadoop Projects. Some of them are given below:

i) Common:

A set of components and interfaces for distributed filesystems and general I/O (serialization, Java RPC, persistent data structures).

ii) Avro:

A serialization system for efficient, cross-language RPC, and persistent data storage.

iii) MapReduce:

A distributed data processing model and execution environment that runs on large clusters of commodity machines.

Apache Hadoop Ecosystem – Contd.

iv) HDFS:

A distributed filesystem that runs on large clusters of commodity machines.

v) Pig:

A data flow language and execution environment for exploring very large datasets. Pig runs on HDFS and MapReduce clusters.

vi) Hive:

A distributed data warehouse. Hive manages data stored in HDFS and provides a query language based on SQL (and which is translated by the runtime engine to MapReduce jobs) for querying the data.

vii) HBase:

A distributed, column-oriented database. HBase uses HDFS for its underlying storage, and supports both batch-style computations using MapReduce and point queries (random reads).

Apache Hadoop Ecosystem – Contd.

viii) ZooKeeper:

A distributed, highly available coordination service. ZooKeeper provides primitives such as distributed locks that can be used for building distributed applications.

ix) Sqoop:

A tool for efficiently moving data between relational databases and HDFS.

x) Oozie:

A Service for running and scheduling workflows of Hadoop job (including MapReduce, Pig, Hive and Sqoop jobs).

xi) Flume:

A distributed, reliable and available service for efficiently collecting, aggregating and moving large amounts of log data.

Apache Hadoop Ecosystem – Contd.

Components of Hadoop Ecosystem

Let us start with the first component HDFS of Hadoop Ecosystem.

HDFS (HADOOP DISTRIBUTED FILE SYSTEM)

- HDFS is a storage layer for Hadoop.
- HDFS is suitable for distributed storage and processing, that is, while the data is being stored, it first gets distributed and then it is processed.
- HDFS provides Streaming access to file system data.
- HDFS provides file permission and authentication.
- HDFS uses a command line interface to interact with Hadoop.

So what stores data in HDFS? It is the HBase which stores data in HDFS.

Apache Hadoop Ecosystem – Contd.

HBase

- HBase is a NoSQL database or non-relational database.
- HBase is important and mainly used when you need random, real-time, read or write access to your Big Data.
- It provides support to a high volume of data and high throughput.
- In an HBase, a table can have thousands of columns.

We discussed how data is distributed and stored. Now, let us understand how this data is ingested or transferred to HDFS. Sqoop does exactly this.

Apache Hadoop Ecosystem – Contd.

What is Sqoop?

- Sqoop is a tool designed to transfer data between Hadoop and relational database servers.
- It is used to import data from relational databases (such as Oracle and MySQL) to HDFS and export data from HDFS to relational databases.

If you want to ingest event data such as streaming data, sensor data, or log files, then you can use Flume. We will look at the flume in the next section.

Apache Hadoop Ecosystem – Contd.

Flume

- Flume is a distributed service that collects event data and transfers it to HDFS.
- It is ideally suited for event data from multiple systems.

After the data is transferred into the HDFS, it is processed. One of the frameworks that process data is Spark.

What is Spark?

- Spark is an open source cluster computing framework.
- It provides up to 100 times faster performance for a few applications with in-memory primitives as compared to the two-stage disk-based MapReduce paradigm of Hadoop.
- Spark can run in the Hadoop cluster and process data in HDFS.
- It also supports a wide variety of workload, which includes Machine learning, Business intelligence, Streaming, and Batch processing.

Apache Hadoop Ecosystem – Contd.

Spark has the following major components:

- Spark Core and Resilient Distributed datasets or RDD
- Spark SQL
- Spark streaming
- Machine learning library or Mlib
- Graphx.

Spark is now widely used, and you will learn more about it in subsequent lessons.

Apache Hadoop Ecosystem – Contd.

Hadoop MapReduce

- Hadoop MapReduce is the other framework that processes data.
- It is the original Hadoop processing engine, which is primarily Java-based.
- It is based on the map and reduces programming model.
- Many tools such as Hive and Pig are built on a map-reduce model.
- It has an extensive and mature fault tolerance built into the framework.
- It is still very commonly used but losing ground to Spark.

After the data is processed, it is analyzed. It can be done by an open-source high-level data flow system called Pig. It is used mainly for analytics.

Apache Hadoop Ecosystem – Contd.

Pig

- Pig converts its scripts to Map and Reduce code, thereby saving the user from writing complex MapReduce programs.
- Ad-hoc queries like Filter and Join, which are difficult to perform in MapReduce, can be easily done using Pig.
- You can also use Impala to analyze data.
- It is an open-source high-performance SQL engine, which runs on the Hadoop cluster.
- It is ideal for interactive analysis and has very low latency which can be measured in milliseconds.

Apache Hadoop Ecosystem – Contd.

Impala

- Impala supports a dialect of SQL, so data in HDFS is modeled as a database table.
- You can also perform data analysis using HIVE. It is an abstraction layer on top of Hadoop.
- It is very similar to Impala. However, it is preferred for data processing and Extract Transform Load, also known as ETL, operations.
- Impala is preferred for ad-hoc queries.

Apache Hadoop Ecosystem – Contd.

HIVE

- HIVE executes queries using MapReduce; however, a user need not write any code in low-level MapReduce.
- Hive is suitable for structured data. After the data is analyzed, it is ready for the users to access.

Now that we know what HIVE does, we will discuss what supports the search of data. Data search is done using Cloudera Search.

Apache Hadoop Ecosystem – Contd.

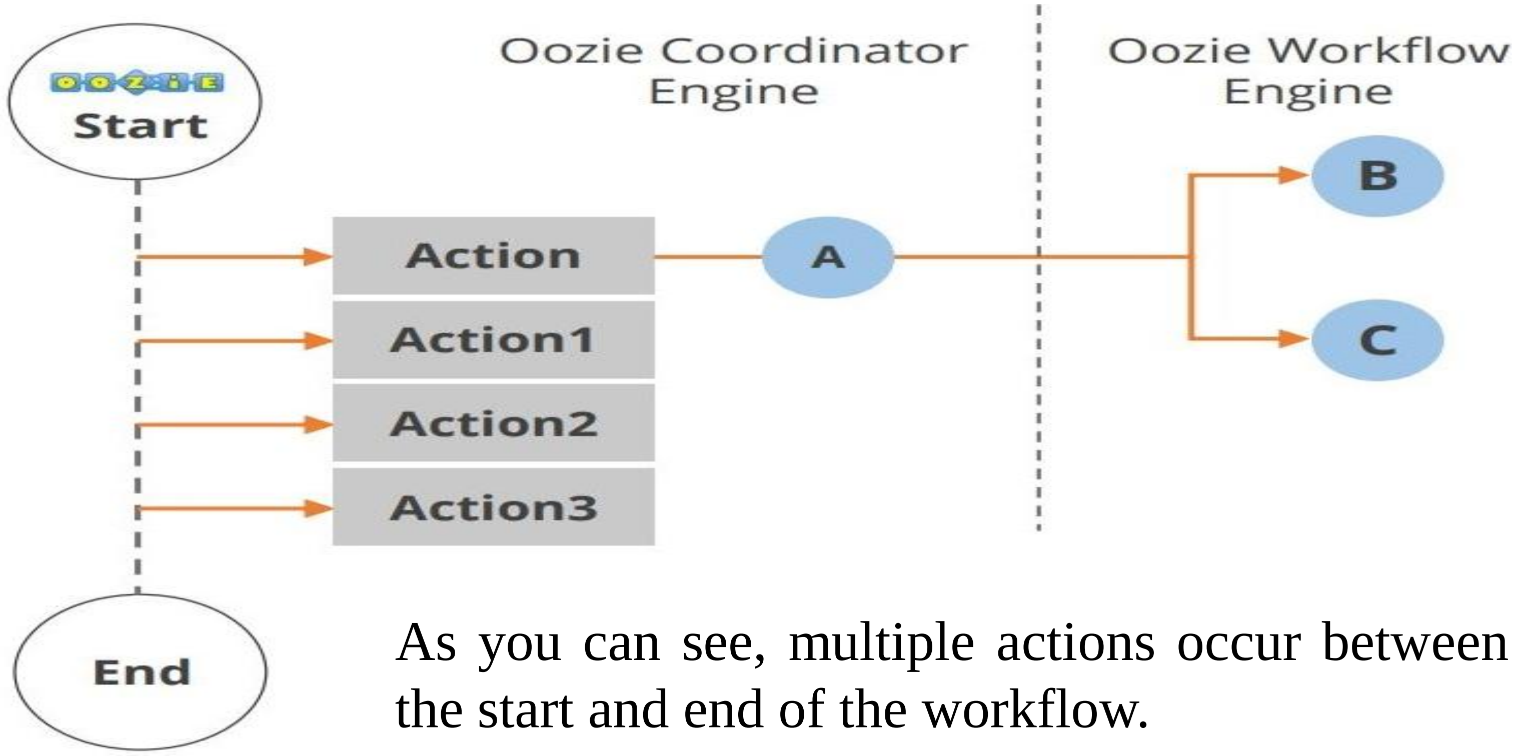
Cloudera Search

- Search is one of Cloudera's near-real-time access products. It enables non-technical users to search and explore data stored in or ingested into Hadoop and HBase.
- Users do not need SQL or programming skills to use Cloudera Search because it provides a simple, full-text interface for searching.
- Another benefit of Cloudera Search compared to stand-alone search solutions is the fully integrated data processing platform.
- Cloudera Search uses the flexible, scalable, and robust storage system included with CDH or Cloudera Distribution, including Hadoop. This eliminates the need to move large datasets across infrastructures to address business tasks.
- Hadoop jobs such as MapReduce, Pig, Hive, and Sqoop have workflows.

Apache Hadoop Ecosystem – Contd.

Oozie

- Oozie is a workflow or coordination system that you can use to manage Hadoop jobs.
- The Oozie application lifecycle is shown in the diagram below.



As you can see, multiple actions occur between the start and end of the workflow.

Apache Hadoop Ecosystem – Contd.

Hue

- Hue is an acronym for Hadoop User Experience. It is an open-source web interface for Hadoop. You can perform the following operations using Hue:
- Upload and browse data
- Query a table in HIVE and Impala
- Run Spark and Pig jobs and workflows Search data
- All-in-all, Hue makes Hadoop easier to use.
- It also provides SQL editor for HIVE, Impala, MySQL, Oracle, PostgreSQL, SparkSQL, and Solr SQL.

After this brief overview of the above components of the Hadoop ecosystem, we will now discuss how these components work together to process Big Data.

Apache Hadoop Ecosystem – Contd.

Stages of Big Data processing

There are four stages of Big Data processing: Ingest, Processing, Analyze, Access. Let us look at them in detail.

- **Ingest**

  The first stage of Big Data processing is Ingest. The data is ingested or transferred to Hadoop from various sources such as relational databases, systems, or local files. Sqoop transfers data from RDBMS to HDFS, whereas Flume transfers event data.

- **Processing**

  The second stage is Processing. In this stage, the data is stored and processed. The data is stored in the distributed file system, HDFS, and the NoSQL distributed data, HBase. Spark and MapReduce perform the data processing.

Apache Hadoop Ecosystem – Contd.

- **Analyze**

  The third stage is Analyze. Here, the data is analyzed by processing frameworks such as Pig, Hive, and Impala.

  Pig converts the data using a map and reduce and then analyzes it. Hive is also based on the map and reduce programming and is most suitable for structured data.

- **Access**

  The fourth stage is Access, which is performed by tools such as Hue and Cloudera Search. In this stage, the analyzed data can be accessed by users.

Hadoop Distributed File System

- Hadoop File System was developed using distributed file system design.
- It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault-tolerant and designed using low-cost hardware.

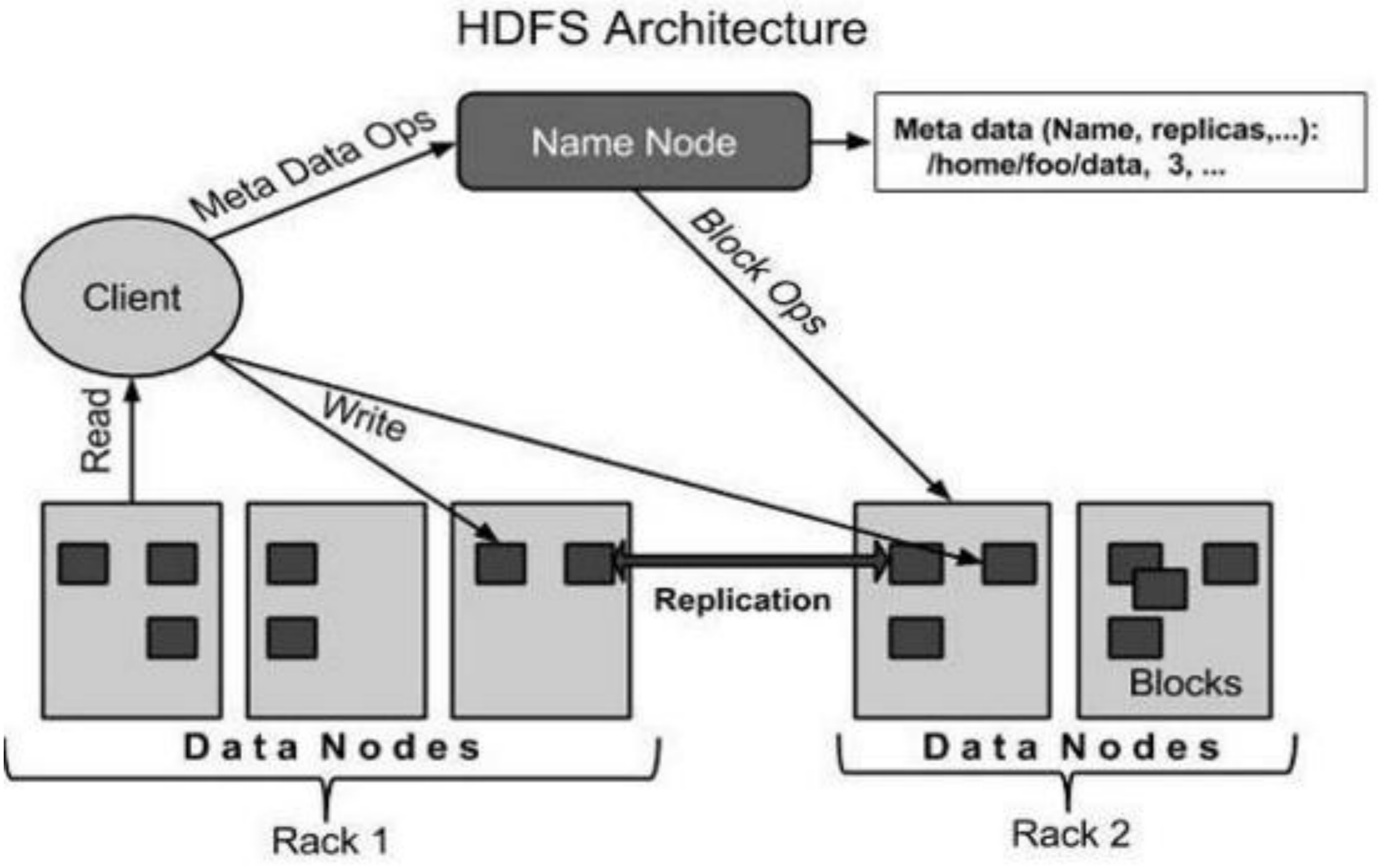
HDFS – Contd.

- HDFS holds very large amount of data and provides easier access.
- To store such huge data, the files are stored across multiple machines.
- These files are stored in redundant fashion to rescue the system from possible data losses in case of failure.
- HDFS also makes applications available to parallel processing.

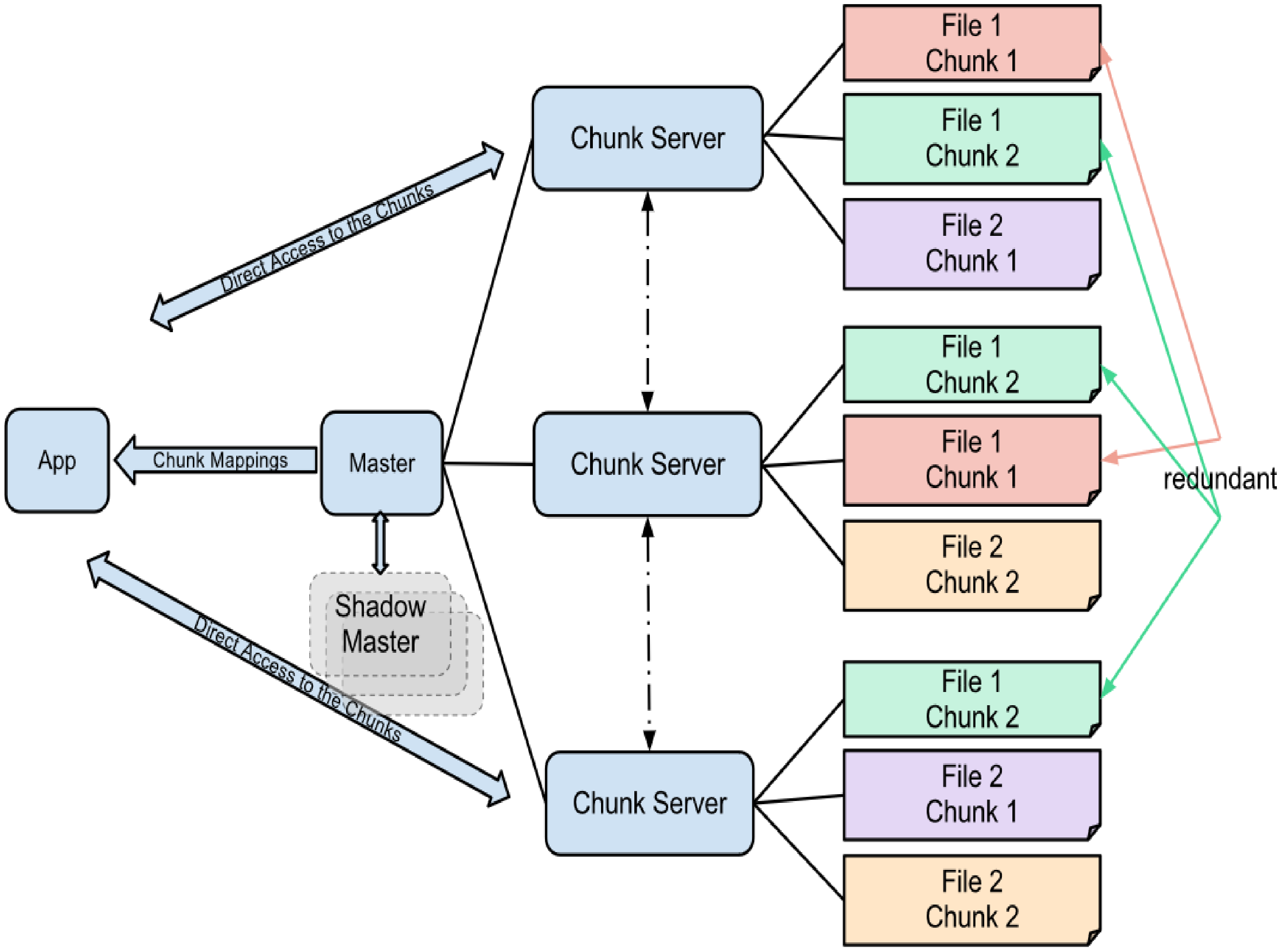
Features of HDFS

- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of Name Node and Data Node help users to easily check the status of cluster.
- Streaming access to file system data.
- HDFS provides file permissions and authentication.

HDFS Architecture



HDFS Architecture Contrast with GFS



HDFS Architecture – Contd.

HDFS follows the master-slave architecture and it has the following elements.

NameNode:

- The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks:
- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

HDFS Architecture – Contd.

DataNode:

- The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.
- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

HDFS Architecture – Contd.

Block:

- Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

HDFS Architecture – Contd.

Goals of HDFS:

- **Fault detection and recovery** : Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.
- **Huge datasets** : HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.
- **Hardware at data** : A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.

The building blocks of Hadoop

In previous discussions, we seen the concepts of distributed storage and distributed computations.

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:

The building blocks of Hadoop – Contd.

- NameNode
- DataNode
- Secondary NameNode
- JobTracker
- TaskTracker

The building blocks of Hadoop – Contd.

NameNode

- Let’s begin with arguably the most vital of the Hadoop daemons—the NameNode. Hadoop employs a master/slave architecture for both distributed storage and distributed computation.
- The distributed storage system is called the *Hadoop File System*, or 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 file system.

The building blocks of Hadoop – Contd.

- 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.

The building blocks of Hadoop – Contd.

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 file system.
- 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.

The building blocks of Hadoop – Contd.

- The following Figure illustrates the roles of the NameNode and DataNodes. In this figure, we show two data files, one at /user/chuck/data1 and another at /user/james/data2.
- The data1 file takes up three blocks, which we denote 1, 2, and 3, and the data2 file consists of blocks 4 and 5. The content of the files are distributed among the DataNodes.
- In this illustration, each block has three replicas. For example, block 1 (used for data1) is replicated over the three rightmost DataNodes. This ensures that if any one DataNode crashes or becomes inaccessible over the network, you'll still be able to read the files.

The building blocks of Hadoop – Contd.

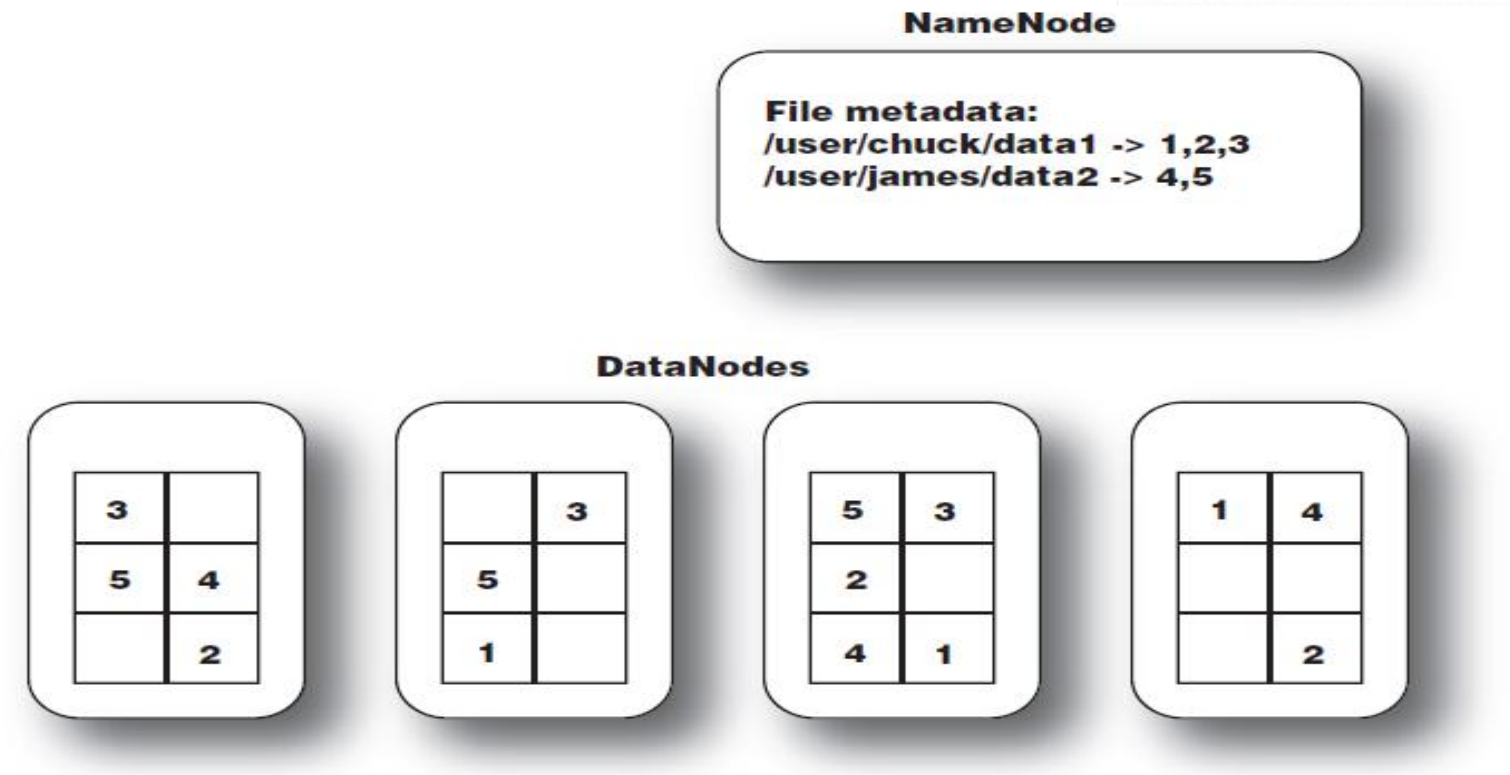


Fig. NameNode / DataNode interaction in HDFS. The NameNode keeps track of the file metadata—which files are in the system and how each file is broken down into blocks. The DataNodes provide backup store of the blocks and constantly report to the NameNode to keep the metadata current.

The building blocks of Hadoop – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

- 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.

The building blocks of Hadoop – Contd.

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.

The building blocks of Hadoop – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

- 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.

The building blocks of Hadoop – Contd.

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.

The building blocks of Hadoop – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

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. The following Figure shows this interaction.

The building blocks of Hadoop – Contd.

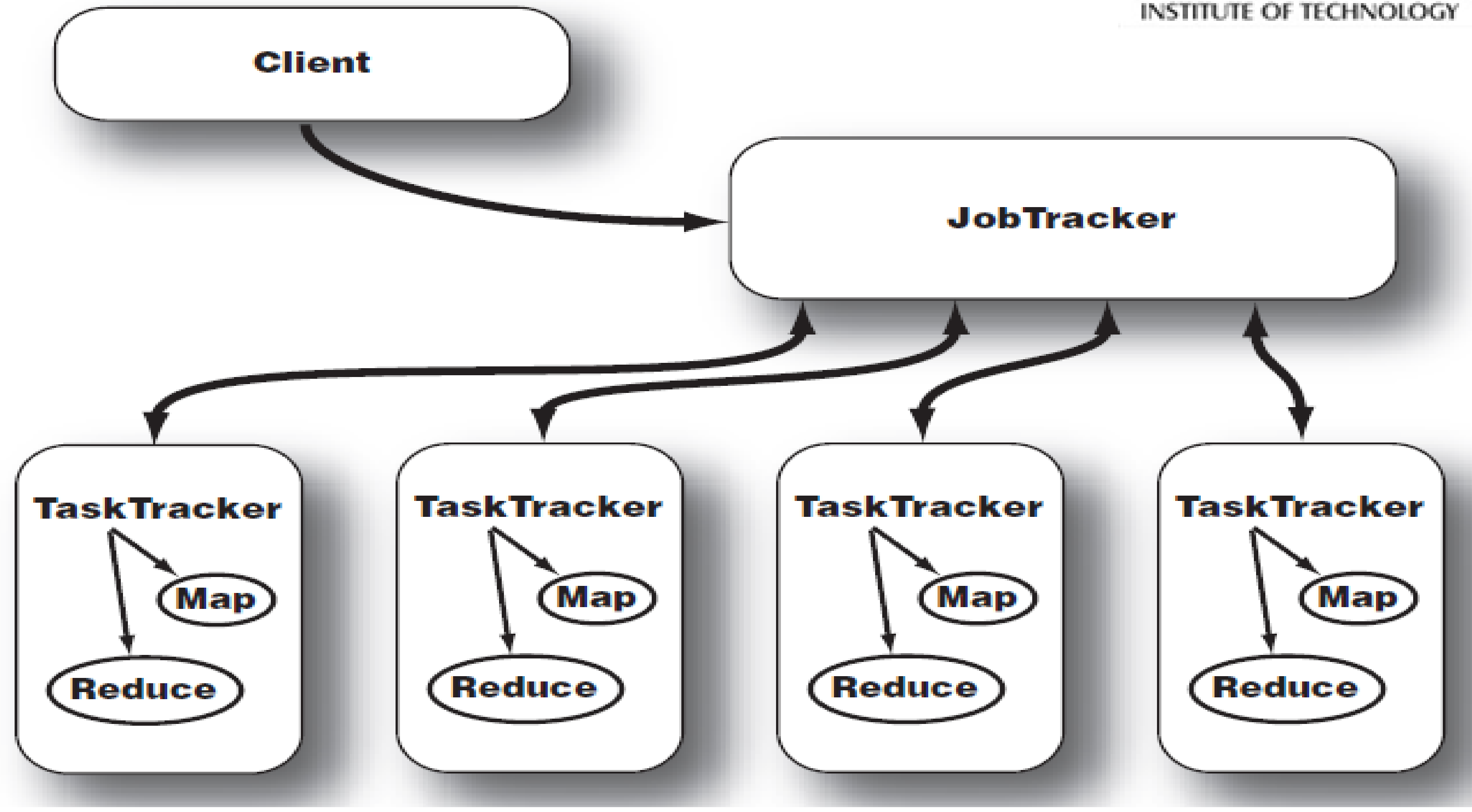


Fig. JobTracker and TaskTracker interaction. After a client calls the JobTracker to begin a data processing job, the JobTracker partitions the work and assigns different map and reduce tasks to each TaskTracker in the cluster.

The building blocks of Hadoop – Contd.

- 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.

The building blocks of Hadoop – Contd.

Having covered each of the Hadoop daemons, we depict the topology of one typical Hadoop cluster in following figure.

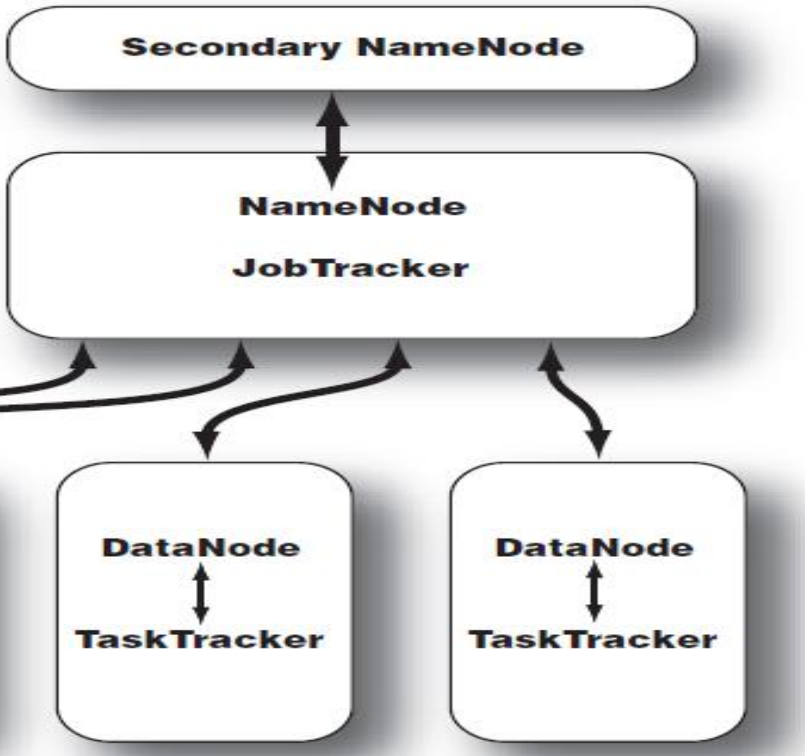


Fig. Topology of a typical Hadoop cluster. It's a master/slave architecture in which the NameNode and JobTracker are masters and the DataNodes and TaskTrackers are slaves.

The building blocks of Hadoop – Contd.

- This topology features a master node running the NameNode and JobTracker daemons and a standalone node with the SNN in case the master node fails.
- For small clusters, the SNN can reside on one of the slave nodes. On the other hand, for large clusters, separate the NameNode and JobTracker on two machines.
- The slave machines each host a DataNode and TaskTracker, for running tasks on the same node where their data is stored.

Introducing and Configuring Hadoop cluster

- Apache Hadoop is an open source framework for storing and distributed batch processing of huge datasets on clusters of commodity hardware.
- Hadoop can be used on a single machine (Standalone Mode) as well as on a cluster of machines (Distributed Mode – Pseudo & Fully).
- One of the striking features of Hadoop is that it efficiently distributes large amounts of work across a cluster of machines / commodity hardware.

Introducing and Configuring Hadoop cluster – Contd.

Configuring Hadoop Cluster:

Before configuring Hadoop Cluster, it is important to understand that Hadoop can be run in any of the following 3 modes:

Standalone Mode

Pseudo-Distributed Mode

Fully Distributed Mode

Introducing and Configuring Hadoop cluster – Contd.

Standalone Mode:

In standalone mode, we will configure Hadoop on a single machine (e.g. an Ubuntu machine on the host VM). The configuration in standalone mode is quite straightforward and does not require major changes.

Pseudo-Distributed Mode:

In a pseudo distributed environment, we will configure more than one machine, one of these to act as a master and the rest as slave machines/node. In addition we will have more than one Ubuntu machine playing on the host VM.

Introducing and Configuring Hadoop cluster – Contd.

Fully Distributed Mode:

Fully Distributed Mode is quite similar to a pseudo distributed environment with the exception that instead of VM the machines/node will be on a real distributed environment.

To configure Hadoop, we need some pre-requisites which are given below:

Introducing and Configuring Hadoop cluster – Contd.

- Hadoop is supported by GNU/Linux platform and its flavors. Therefore, we have to install a Linux operating system for setting up Hadoop environment. In case you have an OS other than Linux, you can install a Virtualbox software in it and have Linux inside the Virtualbox.
- Hadoop requires Java 1.5+ installation. However, using Java 1.6 is recommended for running Hadoop. It can be run on both Windows & Unix but Linux/Unix best support the production environment. Working with Hadoop on Windows also requires Cygwin installation.

CDH Installation

Introducing and Configuring Hadoop cluster – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Basic Hadoop Admin Commands

The ~/hadoop/bin directory contains some scripts used to launch Hadoop DFS and Hadoop Map/Reduce daemons. These are:

startall.sh	– Starts all Hadoop daemons, the namenode, datanodes, the jobtracker and tasktrackers.
stopall.sh	– Stops all Hadoop daemons.
startmapred.sh	– Starts the Hadoop Map/Reduce daemons, the jobtracker and tasktrackers.
stopmapred.sh	– Stops the Hadoop Map/Reduce daemons.
startdfs.sh	– Starts the Hadoop DFS daemons, the namenode and datanodes
stopdfs.sh	– Stops the Hadoop DFS daemons

Live CDH Installation

Introducing and Configuring Hadoop cluster – Contd.

Introducing and Configuring Hadoop cluster

Local

Pseudo-distributed mode

Fully Distributed mode

We all know that Apache Hadoop is an open source framework that allows distributed processing of large sets of data set across different clusters using simple programming. Hadoop has the ability to scale up to thousands of computers from a single server. Thus in these conditions installation of Hadoop becomes most critical. We can install Hadoop in three different modes:

Standalone mode - Single Node Cluster

Pseudo distributed mode - Single Node Cluster

Distributed mode. - Multi Node Cluster

Introducing and Configuring Hadoop cluster – Contd.

Purpose for Different Installation Modes

When Apache Hadoop is used in a production environment, multiple server nodes are used for distributed computing. But for understanding the basics and playing around with Hadoop, single node installation is sufficient. There is another mode known as 'pseudo distributed' mode. This mode is used to simulate the multi node environment on a single server.

We will discuss how to install Hadoop on Ubuntu Linux. In any mode, the system should have java version 1.6.x installed on it.

Introducing and Configuring Hadoop cluster – Contd.

Standalone Mode Installation

Now, let us check the standalone mode installation process by following the steps mentioned below.

Install Java

Java (JDK Version 1.6.x) either from Sun/Oracle or Open Java is required.

Step 1 - If you are not able to switch to OpenJDK instead of using proprietary Sun JDK/JRE, install sun-java6 from Canonical Partner Repository by using the following command.

Note: The *Canonical Partner Repository* contains free of cost closed source third party software. But the Canonical does not have access to the source code instead they just package and test it.

Introducing and Configuring Hadoop cluster – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Add the canonical partner to the apt repositories using -

$ sudo add-apt-repository "deb http://archive.canonical.com/lucid partner"

Step 2 - Update the source list.

$ sudo apt-get update

Step 3 - Install JDK version 1.6.x from Sun/Oracle.

$ sudo apt-get install sun-java6-jdk

Step 4 - Once JDK installation is over make sure that it is correctly setup using - version

1.6.x from Sun/Oracle. user@ubuntu:~# java -version java version "1.6.0_45"

Java(TM) SE Runtime Environment (build 1.6.0_45-b02)

Java HotSpot(TM) Client VM (build 16.4-b01, mixed mode, sharing)

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Introducing and Configuring Hadoop cluster – Contd.

Add Hadoop User

Step 5 - Add a dedicated Hadoop unix user into you system as under to isolate this installation from other software -

```
$ sudo adduser hadoop_admin
```

Download the Hadoop binary and install

Step 6 - Download Apache Hadoop from the apache web site. Hadoop comes in the form of tar-gx format. Copy this binary into the /usr/local/installables folder. The folder - installables should be created first under /usr/local before this step. Now run the following commands as sudo

```
$ cd /usr/local/installables

$ sudo tar xzf hadoop-0.20.2.tar.gz

$ sudo chown -R hadoop_admin /usr/local/hadoop-0.20.2
```

Introducing and Configuring Hadoop cluster – Contd.

Define env variable - JAVA_HOME

Step 7 - Open the Hadoop configuration file (hadoop-env.sh) in the location -

/usr/local/installables/hadoop-0.20.2/conf/hadoop-env.sh and define the JAVA_HOME as under –

```
export JAVA_HOME=path/where/jdk/is/installed (e.g. /usr/bin/java)
```

Installation in Single mode

Step 8 - Now go to the HADOOP_HOME directory (location where HADOOP is extracted) and run the following command –

```
$ bin/hadoop
```

The following output will be displayed –

```
Usage: hadoop [--config confdir] COMMAND
```

Introducing and Configuring Hadoop cluster – Contd.

Some of the COMMAND options are mentioned below. There are other options available and can be checked using the command mentioned above.

namenode -format	format the DFS filesystem
secondarynamenode	run the DFS secondary namenode
namenode	run the DFS namenode
datanode	run a DFS datanode
dfsadmin	run a DFS admin client
mradmin	run a Map-Reduce admin client
fsck	run a DFS filesystem checking utility

The above output indicates that Standalone installation is completed successfully. Now you can run the sample examples of your choice by calling –

```
$ bin/hadoop jar hadoop-*-examples.jar <NAME> <PARAMS>
```

Introducing and Configuring Hadoop cluster – Contd.

Pseudo Distributed Mode Installation

This is a simulated multi node environment based on a single node server.

Here, the first step required is to configure the SSH in order to access and manage the different nodes. It is mandatory to have the SSH access to the different nodes. Once the SSH is configured, enabled and is accessible we should start configuring the Hadoop. The following configuration files need to be modified:

conf/core-site.xml

conf/hdfs-site.xml

conf/mapred.xml

Open the all the configuration files in vi editor and update the configuration.

Introducing and Configuring Hadoop cluster – Contd.

Configure core-site.xml file:

```
$ vi conf/core-site.xml

<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/tmp/hadoop-${user.name}</value>
</property>
</configuration>
```

Introducing and Configuring Hadoop cluster – Contd.

Configure hdfs-site.xml file:

```
$ vi conf/hdfs-site.xml

<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
```

Introducing and Configuring Hadoop cluster – Contd.

Configure mapred.xml file:

```
$ vi conf/mapred.xml

<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:9001</value>
</property>
</configuration>
```

Introducing and Configuring Hadoop cluster – Contd.

Once these changes are done, we need to format the name node by using the following command. The command prompt will show all the messages one after another and finally success message.

```
$ bin/hadoop namenode –format
```

Our setup is done for pseudo distributed node. Let's now **start the single node cluster** by using the following command. It will again show some set of messages on the command prompt and start the server process.

```
$ /bin/start-all.sh
```

Introducing and Configuring Hadoop cluster – Contd.

Now we should check the status of Hadoop process by executing the jps command as shown below. It will show all the running processes.

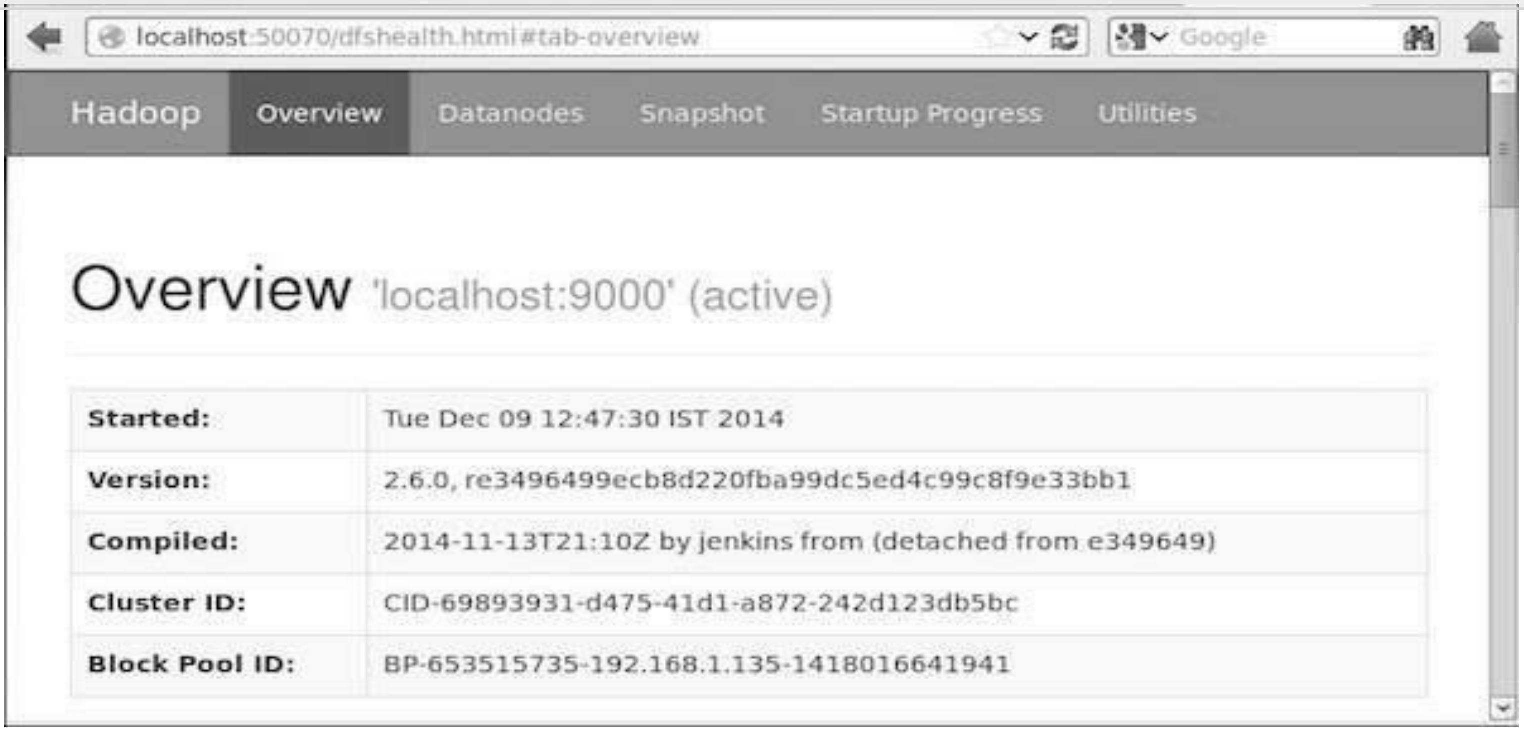
```
$ jps
14799 NameNode
14977 SecondaryNameNode
15183 DataNode
15596 JobTracker
15897 TaskTracker
```

Introducing and Configuring Hadoop cluster – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Accessing Hadoop on Browser:

The default port number to access Hadoop is 50070. Use the following url to get Hadoop services on browser: http://localhost:50070/



Overview 'localhost:9000' (active)

Started:	Tue Dec 09 12:47:30 IST 2014
Version:	2.6.0, re3496499ecb8d220fba99dc5ed4c99c8f9e33bb1
Compiled:	2014-11-13T21:10Z by jenkins from (detached from e349649)
Cluster ID:	CID-69893931-d475-41d1-a872-242d123db5bc
Block Pool ID:	BP-653515735-192.168.1.135-1418016641941

Introducing and Configuring Hadoop cluster – Contd.

Stopping the Single node Cluster:

We can stop the single node cluster by using the following command.

The command prompt will display all the stopping processes.

```
$ bin/stop-all.sh

stopping jobtracker
localhost: stopping tasktracker
stopping namenode
localhost: stopping datanode
localhost: stopping secondarynamenode
```

Introducing and Configuring Hadoop cluster – Contd.

Fully Distributed Mode Installation

Compatibility Requirements

S.No.	Category	Supported
1	Languages	Java, Python, Perl, Ruby etc.
2	Operating System	Linux (Server Deployment) Mostly preferred, Windows (Development only), Solaris.
3	Hardware	32 bit Linux (64 bit for large deployment)

Introducing and Configuring Hadoop cluster – Contd.

Installation Items

Note: Both Items are required to be installed on Namenode and Datanode machines

S.No.	Item	Version
1	**jdk-6u25-linux-i586.bin**	**Java 1.6 or higher**
2	**hadoop-0.20.2-cdh3u0.tar.gz**	**Hadoop 0.20.2**

Introducing and Configuring Hadoop cluster – Contd.

Installation Requirements

S.No.	Requirement	Reason
1	Operating system – Linux recommended for server deployment (Production env.)	
2	Language – Java 1.6 or higher	
3	Ram – at least 4 GB/node	
4	Hard disk – at least 1 TB	For namenode machine.
5	Should have root credentials	For changing some system files you need admin permissions.

Introducing and Configuring Hadoop cluster – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

High Level Steps

1	**Binding IP address with the host name under /etc/hosts**
2	**Setting passwordless SSH**
3	**Installing Java**
4	**Installing Hadoop**
5	**Setting JAVA HOME and HADOOP HOME variables**
6	**Updating .bash_profile file for hadoop**

Introducing and Configuring Hadoop cluster – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

High Level Steps – Contd.

7	Creating required folders for namenode and datanode
8	Configuring the .xml files
9	Setting the masters and slaves in all the machines
10	Formatting the namenode
11	Starting the Dfs services and mapred services
12	Stopping all services

Introducing and Configuring Hadoop cluster – Contd.

Before we start the distributed mode installation, we must ensure that we have the pseudo distributed setup done and we have at least two machines, one acting as master and the other acting as a slave.

Now we run the following commands in sequence.

$ bin/stop-all.sh – Make sure none of the nodes are running

Introducing and Configuring Hadoop cluster – Contd.

Binding IP address with the host names

Before starting the installation of hadoop, first you need to bind the IP address of the machines along with their host names under /etc/hosts file.

First check the hostname of your machine by using following command :

$ hostname

Open /etc/hosts file for binding IP with the hostname

$ vi /etc/hosts

Provide ip & hostname of the all the machines in the cluster e.g: 10.11.22.33 hostname1

10.11.22.34   hostname2

Introducing and Configuring Hadoop cluster – Contd.

Setting Passwordless SSh login

SSH is used to login from one system to another without requiring passwords. This will be required when you run a cluster, it will not prompt you for the password again and again.

First log in on Host1 **(hostname of namenode machine)** as hadoop user and generate a pair of authentication keys. Command is:

```
hadoop@Host1$ ssh-keygen –t rsa
```

Note: Give the hostname which you got in step 5.3.1. Do not enter any passphrase if asked. Now use ssh to create a directory ~/.ssh as user hadoop on Host2 **(Hostname other than namenode machine).**

Introducing and Configuring Hadoop cluster – Contd.

Now we open the two files - conf/master and conf/slaves. The conf/master defines the name nodes of our multi node cluster.

The conf/slaves file lists the hosts where the Hadoop Slave will be running.

Edit the conf/core-site.xml file to have the following entries -

```
<property>
<name>fs.default.name</name>
<value>hdfs://master:54310</value>
</property>
```

Introducing and Configuring Hadoop cluster – Contd.

Edit the conf/mapred-site.xml file to have the following entries -

```
<property>
<name>mapred.job.tracker</name>
<value>hdfs://master:54311</value>
</property>
```

Edit the conf/hdfs-site.xml file to have the following entries

```
<property>
<name>dfs.replication</name>
<value>3</value>
</property>
```

Introducing and Configuring Hadoop cluster – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Edit the conf/mapred-site.xml file to have the following entries.

```
<property>
<name>mapred.local.dir</name>
<value>${hadoop-tmp}/mapred/local</value>
</property>
<property>
<name>mapred.map.tasks</name>
<value>50</value>
</property>
<property>
<name>mapred.reduce.tasks</name>
<value>5</value>
</property>
```

Introducing and Configuring Hadoop cluster – Contd.

Now start the master by using the following command.

```
bin/start-dfs.sh
```

Once started, check the status on the master by using jps command. You should get the following output –

```
14799 NameNode
15314 Jps
16977 secondaryNameNode
```

On the slave, the output should be as shown as:

```
15183 DataNode
15616 Jps
```

Introducing and Configuring Hadoop cluster – Contd.

Now start the MapReduce daemons using the following command.

```
$ bin/start-mapred.sh
```

Once started, check the status on the master by using jps command. You should get the following output:

```
16017 Jps
14799 NameNode
15596 JobTracker
14977 SecondaryNameNode
```

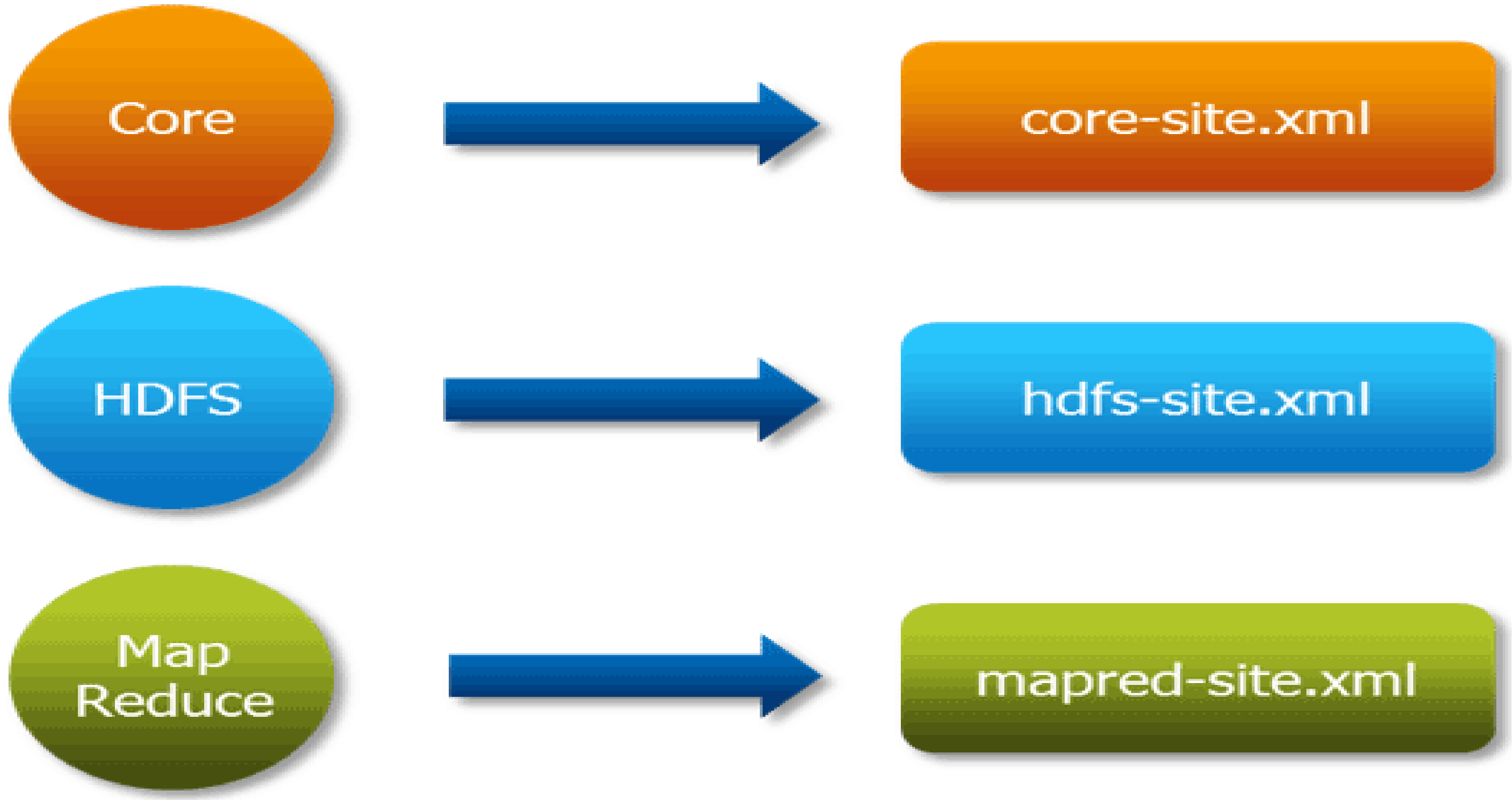
On the slaves, the output should be as shown below.

```
15183 DataNode
15897 TaskTracker
16284 Jps
```

Introducing and Configuring Hadoop cluster – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Configuring XML files

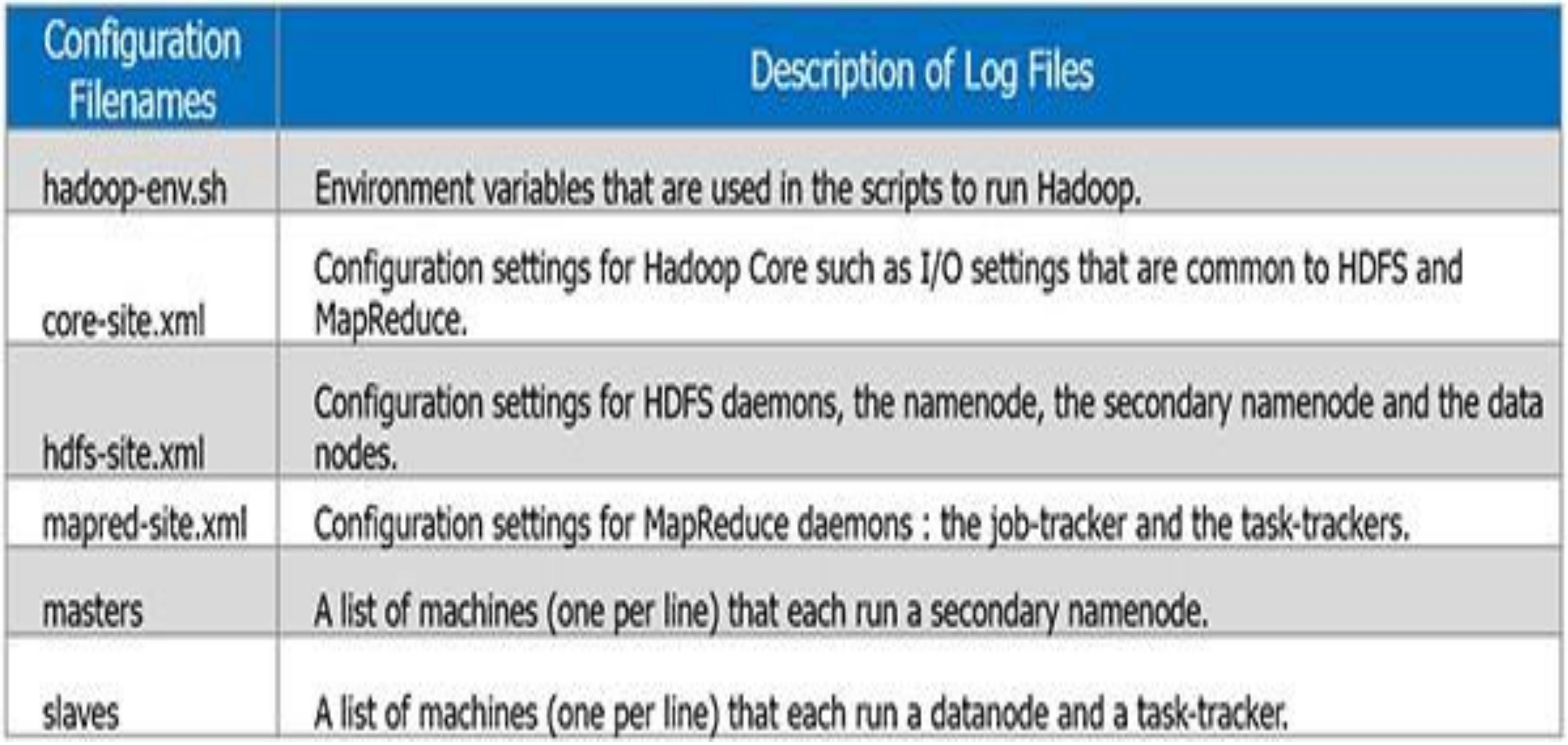


Introducing and Configuring Hadoop cluster – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Hadoop Cluster Configuration Files

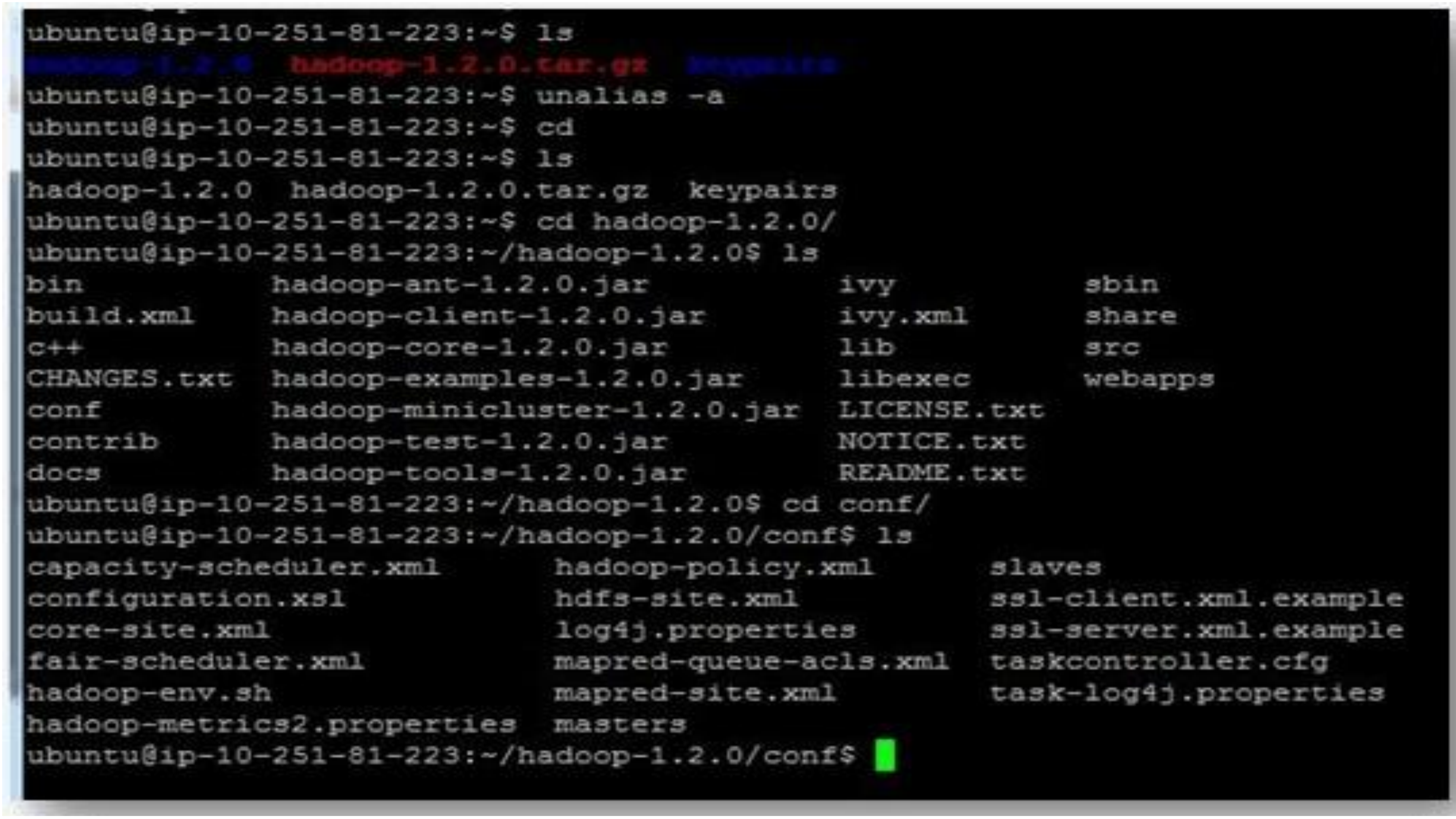
All these files are available under ‘**conf**’ directory of Hadoop installation directory



Configuration Filenames	Description of Log Files
hadoop-env.sh	Environment variables that are used in the scripts to run Hadoop.
core-site.xml	Configuration settings for Hadoop Core such as I/O settings that are common to HDFS and MapReduce.
hdfs-site.xml	Configuration settings for HDFS daemons, the namenode, the secondary namenode and the data nodes.
mapred-site.xml	Configuration settings for MapReduce daemons : the job-tracker and the task-trackers.
masters	A list of machines (one per line) that each run a secondary namenode.
slaves	A list of machines (one per line) that each run a datanode and a task-tracker.

Introducing and Configuring Hadoop cluster – Contd.

Here is a listing of these files in the File System



```
ubuntu@ip-10-251-81-223:~$ ls
hadoop-1.2.0  hadoop-1.2.0.tar.gz  keypairs
ubuntu@ip-10-251-81-223:~$ unalias -a
ubuntu@ip-10-251-81-223:~$ cd
ubuntu@ip-10-251-81-223:~$ ls
hadoop-1.2.0  hadoop-1.2.0.tar.gz  keypairs
ubuntu@ip-10-251-81-223:~$ cd hadoop-1.2.0/
ubuntu@ip-10-251-81-223:~/hadoop-1.2.0$ ls
bin           hadoop-ant-1.2.0.jar          ivy          sbin
build.xml     hadoop-client-1.2.0.jar       ivy.xml      share
c++           hadoop-core-1.2.0.jar         lib          src
CHANGES.txt   hadoop-examples-1.2.0.jar     libexec      webapps
conf          hadoop-minicluster-1.2.0.jar  LICENSE.txt
contrib       hadoop-test-1.2.0.jar         NOTICE.txt
docs          hadoop-tools-1.2.0.jar        README.txt
ubuntu@ip-10-251-81-223:~/hadoop-1.2.0$ cd conf/
ubuntu@ip-10-251-81-223:~/hadoop-1.2.0/conf$ ls
capacity-scheduler.xml      hadoop-policy.xml      slaves
configuration.xsl           hdfs-site.xml          ssl-client.xml.example
core-site.xml               log4j.properties       ssl-server.xml.example
fair-scheduler.xml          mapred-queue-acls.xml  taskcontroller.cfg
hadoop-env.sh               mapred-site.xml        task-log4j.properties
hadoop-metrics2.properties  masters
ubuntu@ip-10-251-81-223:~/hadoop-1.2.0/conf$
```

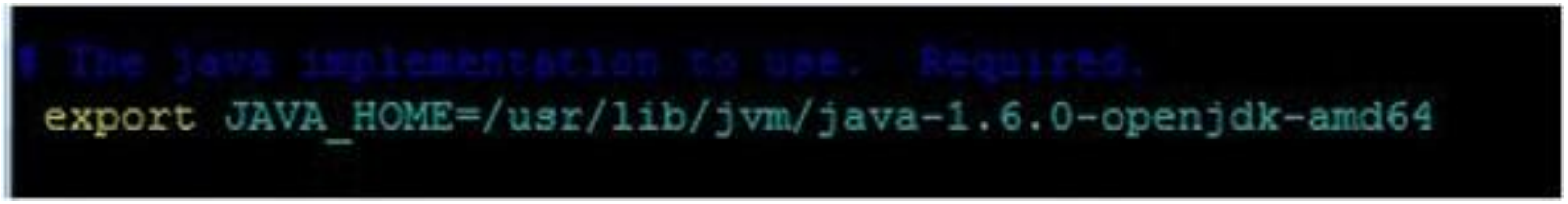
Introducing and Configuring Hadoop cluster – Contd.

Let's look at the files and their usage one by one!

<u>hadoop-env.sh</u>

This file specifies environment variables that affect the JDK used by **Hadoop Daemon** (bin/hadoop).

As Hadoop framework is written in *Java* and uses *Java Runtime environment*, one of the important environment variables for Hadoop daemon is **$JAVA_HOME** in **hadoop-env.sh**. This variable directs Hadoop daemon to the *Java path* in the system.



```
# The java implementation to use.  Required.
export JAVA_HOME=/usr/lib/jvm/java-1.6.0-openjdk-amd64
```

This file is also used for setting another *Hadoop daemon execution environment* such as **heap size (HADOOP_HEAP), hadoop home (HADOOP_HOME), log file location (HADOOP_LOG_DIR),** etc.

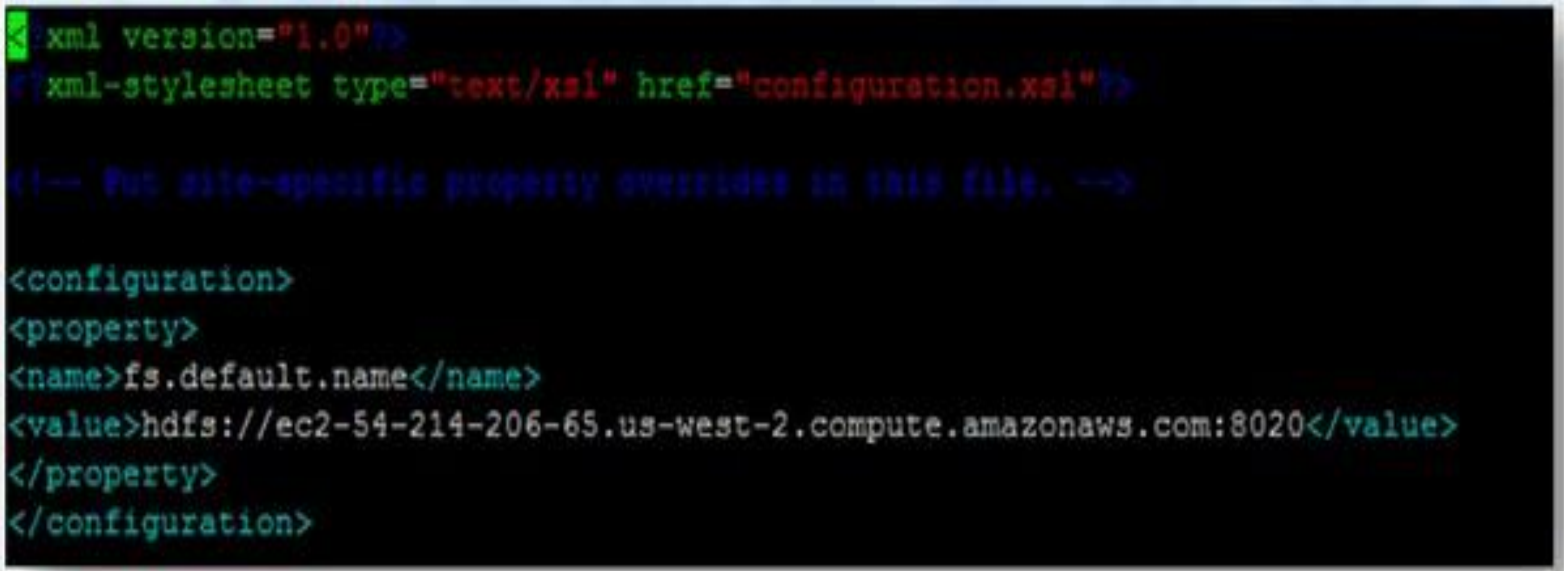
Note: For the simplicity of understanding the cluster setup, we have configured only necessary parameters to start a cluster.

Introducing and Configuring Hadoop cluster – Contd.

The following three files are the important configuration files for the runtime environment settings of a Hadoop cluster.

core-site.xml

This file informs Hadoop daemon where NameNode runs in the cluster. It contains the configuration settings for Hadoop Core such as I/O settings that are common to *HDFS* and *MapReduce*.



```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://ec2-54-214-206-65.us-west-2.compute.amazonaws.com:8020</value>
</property>
</configuration>
```

Introducing and Configuring Hadoop cluster – Contd.

Where *hostname* and *port* are the machine and port on which NameNode daemon runs and listens.

It also informs the Name Node as to which IP and port it should bind.

The commonly used port is **8020** and you can also specify IP address rather than hostname

Introducing and Configuring Hadoop cluster – Contd.

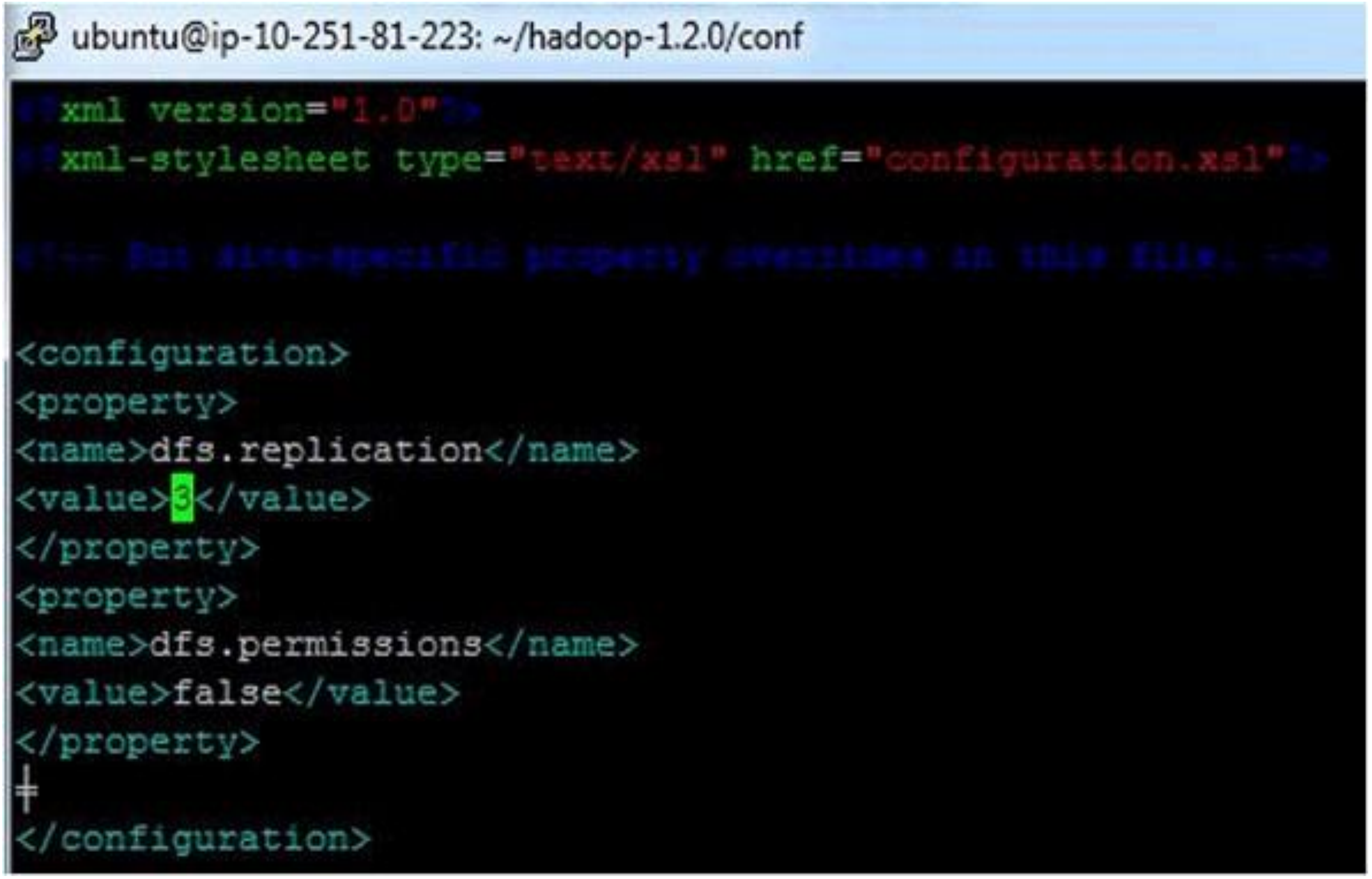
hdfs-site.xml

This file contains the configuration settings for *HDFS daemons; the Name Node, the Secondary Name Node, and the data nodes.*

You can also configure **hdfs-site.xml** to specify default block replication and permission checking on HDFS. The actual number of replications can also be specified when the file is created. The default is used if replication is not specified in create time.

The value “true” for property **‘dfs.permissions’** enables permission checking in HDFS and the value “false” turns off the permission checking. Switching from one parameter value to the other does not change the mode, owner or group of files or directories.

Introducing and Configuring Hadoop cluster – Contd.



ubuntu@ip-10-251-81-223: ~/hadoop-1.2.0/conf

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.replication</name>
<value>3</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>

</configuration>
```

Introducing and Configuring Hadoop cluster – Contd.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

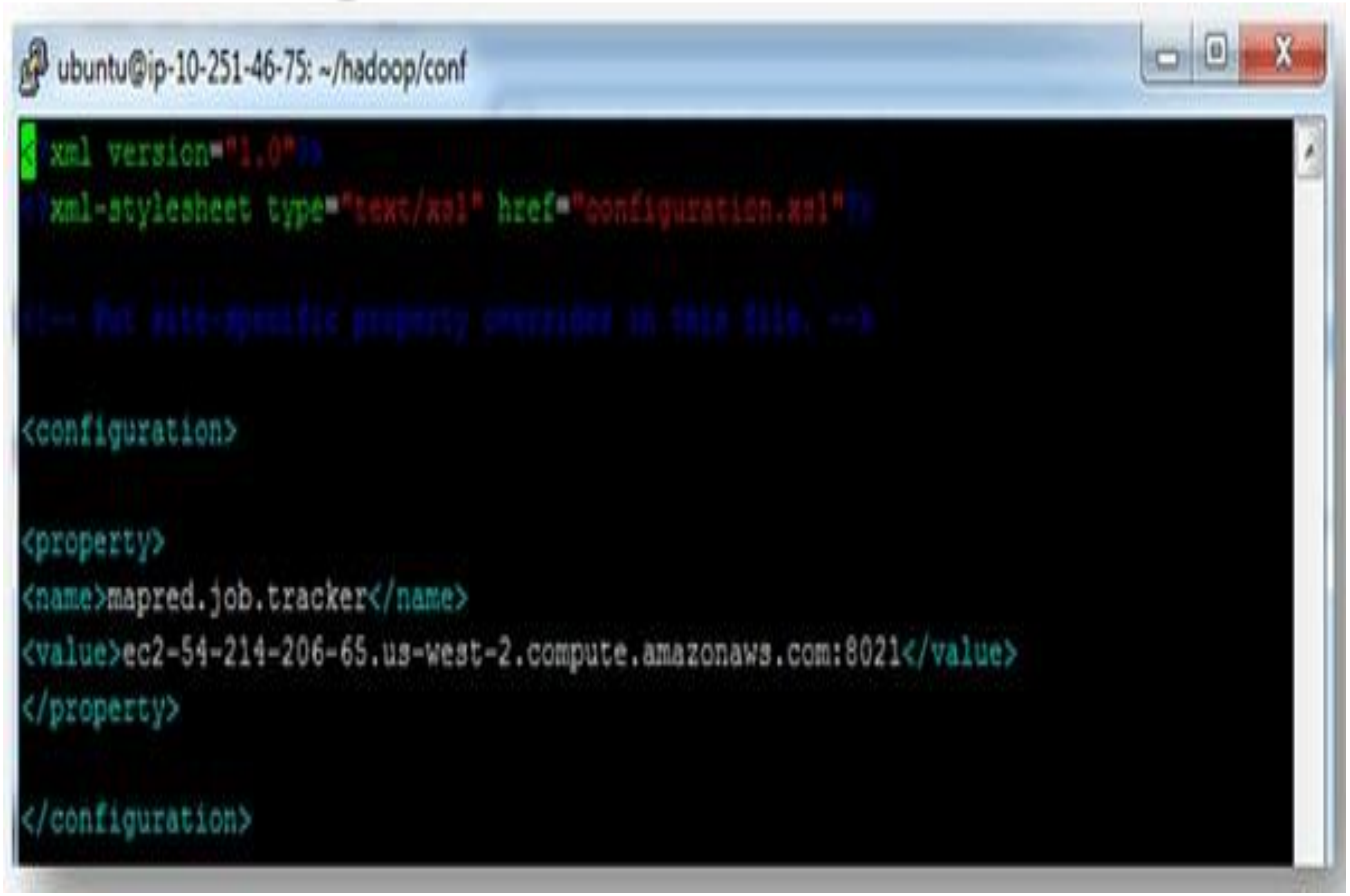
mapred-site.xml

This file contains the configuration settings for MapReduce daemons; *the job tracker and the task-trackers*. The**mapred.job.tracker** parameter is a *hostname* (or IP address) and *port* pair on which the Job Tracker listens for RPC communication.

This parameter specify the location of the Job Tracker to Task Trackers and MapReduce clients.

You can replicate all of the four files explained above to all the Data Nodes and Secondary Namenode. These files can then be configured for any node specific configuration e.g. in case of a different JAVA HOME on one of the Datanodes.

Introducing and Configuring Hadoop cluster – Contd.



ubuntu@ip-10-251-46-75: ~/hadoop/conf

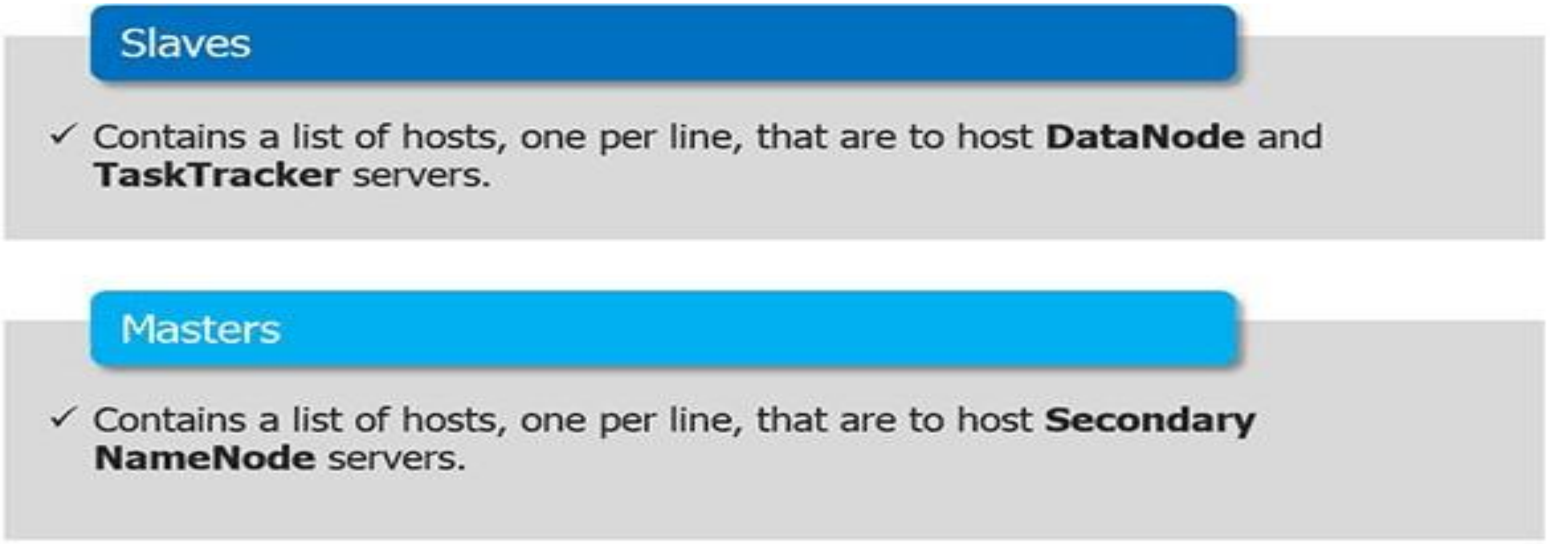
```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>mapred.job.tracker</name>
<value>ec2-54-214-206-65.us-west-2.compute.amazonaws.com:8021</value>
</property>

</configuration>
```

Introducing and Configuring Hadoop cluster – Contd.

The following two file 'masters' and 'slaves' determine the master and salve Nodes in Hadoop cluster.

Masters

This file informs about the Secondary Namenode location to hadoop daemon. The '**masters**' file at Master server contains a hostname Secondary Name Node servers.



Slaves

✓ Contains a list of hosts, one per line, that are to host **DataNode** and **TaskTracker** servers.

Masters

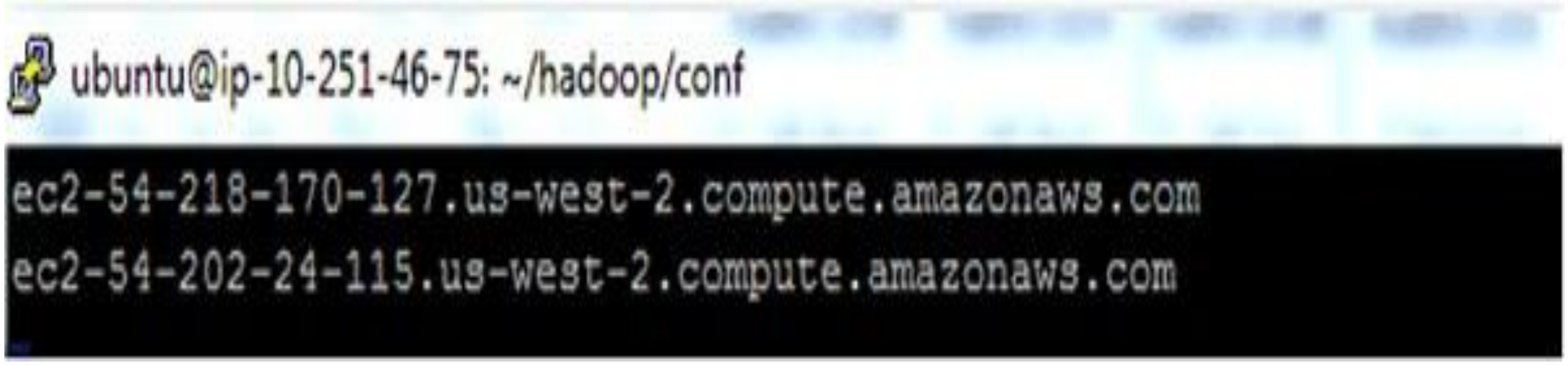
✓ Contains a list of hosts, one per line, that are to host **Secondary NameNode** servers.

The 'masters' file on **Slave Nodes** is blank.

Introducing and Configuring Hadoop cluster – Contd.

Slaves

The ‘**slaves**’ file at Master node contains a list of hosts, one per line, that are to host Data Node and Task Tracker servers.



```
ubuntu@ip-10-251-46-75: ~/hadoop/conf
ec2-54-218-170-127.us-west-2.compute.amazonaws.com
ec2-54-202-24-115.us-west-2.compute.amazonaws.com
```

The ‘**slaves**’ file on Slave node contains the IP address of the slave node. Notice that the ‘slaves’ file at Slave node contains only its own IP address and not of any other Data Nodes in the cluster.

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Now…

We will See

“Live Demo of HDFS File Management Tasks in CDH VM ”