Hadoop and HDFS

MR. Huynh Nam

Outline

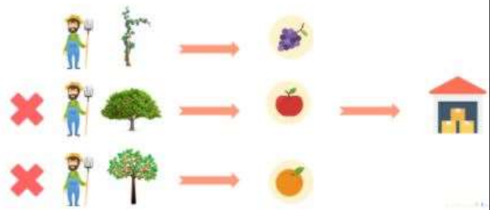
Hadoop Introduction

What is Hadoop

- Hadoop is a framework that uses distributed storage and parallel processing to store and manage big data. It is the software most used by data analysts to handle big data, and its market size continues to grow.
- There are three components of Hadoop:
  - Hadoop HDFS - Hadoop Distributed File System (HDFS) is the storage unit.
  - Hadoop MapReduce - Hadoop MapReduce is the processing unit.
  - Hadoop YARN - Yet Another Resource Negotiator (YARN) is a resource management unit.

Hadoop Through an Analogy - 1

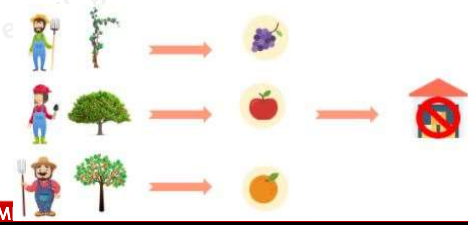
- Introducing Jack, a grape farmer. He harvests the grapes in the fall, stores them in a storage room, and finally sells them in the nearby town. He kept this routing going for years until people began to demand other fruits. This rise in demand led to him growing apples and oranges, in addition to grapes.
- Unfortunately, the whole process turned out to be **time-consuming** and difficult for Jack to do single-handedly.



www.flex.edu.vn

Hadoop Through an Analogy - 2

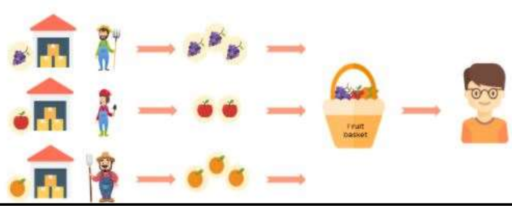
- So, Jack hires two more people to work alongside him. The extra help speeds up the harvesting process as three of them can work simultaneously on different products.
- However, this takes a nasty toll on the storage room, as the storage area becomes a **bottleneck for storing and accessing** all the fruits.



www.flex.edu.vn

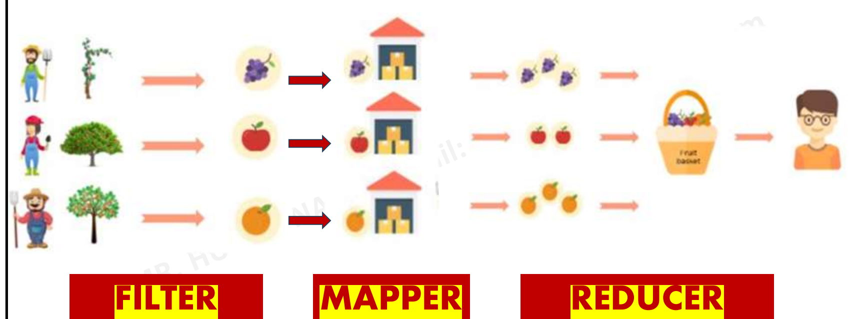
Hadoop Through an Analogy - 3

- Jack thought through this problem and came up with a solution: **give each one separate storage space**. So, when Jack receives an order for a fruit basket, he can complete the order on time as all three can work with their storage area.
- Thanks to Jack's solution, everyone can finish their order on time and with no trouble. Even with sky-high demands, Jack can complete his orders.



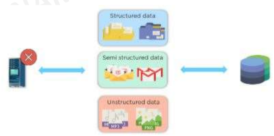
www.flex.edu.vn

Filter – Mapper – Reducer Pipeline



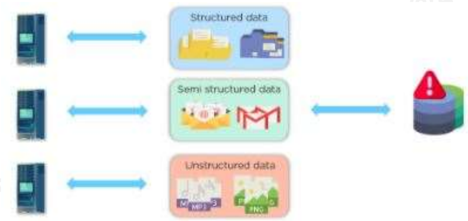
The Rise of Big Data - 1

- Back in the day, there was limited data generation. Hence, storing and processing data was done with a single storage unit and a processor, respectively. In the blink of an eye, data generation increases by leaps and bounds. Not only did it increase in volume but also its variety.
- Therefore, a single processor was incapable of processing high volumes of different varieties of data. Speaking of varieties of data, you can have structured, semi-structured and unstructured data.



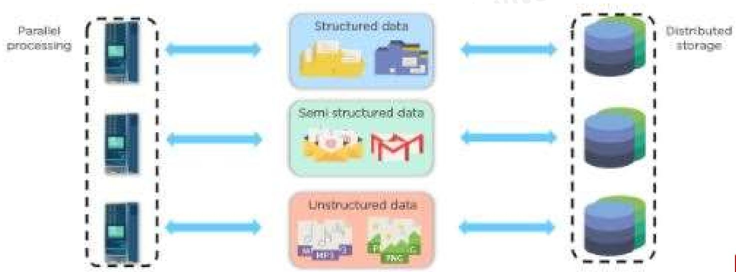
The Rise of Big Data - 2

- Multiple machines help process data parallelly. However, the storage unit became a bottleneck resulting in a network overhead generation



The Rise of Big Data - 3

- To address this issue, the storage unit is distributed amongst each of the processors. The distribution resulted in storing and accessing data efficiently and with no network overheads. As seen below, this method is called parallel processing with distributed storage.



Big Data and Its Challenges

- Big Data refers to the massive amount of data that cannot be stored, processed, and analyzed using traditional ways.
- The main elements of Big Data are:
  - Volume - There is a massive amount of data generated every second.
  - Velocity - The speed at which data is generated, collected, and analyzed
  - Variety - The different types of data: structured, semi-structured, unstructured
  - Value - The ability to turn data into useful insights for your business
  - Veracity - Trustworthiness in terms of quality and accuracy

The main challenges that Big Data faced and the solutions

Challenges	Solution
Single central storage	Distributed storage
Serial processing •One input •One processor •One output	Parallel processing •Multiple inputs •Multiple processors •One output
Lack of ability to process unstructured data	Ability to process every type of data

Why Is Hadoop Important?

- Hadoop is a beneficial technology for data analysts. There are many essential features in Hadoop which make it so important and user-friendly.
- The system is able to store and process enormous amounts of data at an extremely fast rate. A semi-structured, structured and unstructured data set can differ depending on how the data is structured.
- Enhances operational decision-making and batch workloads for historical analysis by supporting real-time analytics.
- Data can be stored by organizations, and it can be filtered for specific analytical uses by processors as needed.
- A large number of nodes can be added to Hadoop as it is scalable, so organizations will be able to pick up more data.
- A protection mechanism prevents applications and data processing from being harmed by hardware failures. Nodes that are down are automatically redirected to other nodes, allowing applications to run without interruption.

Who Uses Hadoop?

- Hadoop is a popular big data tool, used by many companies worldwide. Here's a brief sample of successful Hadoop users:
  - British Airways
  - Uber
  - The Bank of Scotland
  - Netflix
  - The National Security Agency (NSA), of the United States
  - The UK's Royal Mail system
  - Expedia
  - Twitter

Components of Hadoop

- Hadoop is a framework that uses distributed storage and parallel processing to store and manage Big Data. It is the most commonly used software to handle Big Data.
- There are three components of Hadoop.
  - Hadoop HDFS - Hadoop Distributed File System (HDFS) is the storage unit of Hadoop.
  - Hadoop MapReduce - Hadoop MapReduce is the processing unit of Hadoop.
  - Hadoop YARN - Hadoop YARN is a resource management unit of Hadoop.

Hadoop HDFS

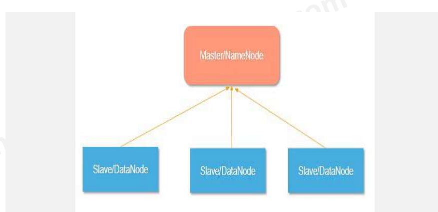
- Data is stored in a distributed manner in HDFS. There are two components of HDFS - name node and data node. While there is **only one name node**, there can be **multiple data nodes**.
- HDFS is specially designed for storing huge datasets in commodity hardware. An enterprise version of a server costs roughly $10,000 per terabyte for the full processor. In case you need to buy 100 of these enterprise version servers, it will go up to a million dollars.
- Hadoop enables you to use commodity machines as your data nodes. This way, you don't have to spend millions of dollars just on your data nodes. However, **the name node is always an enterprise server.**

Features of HDFS

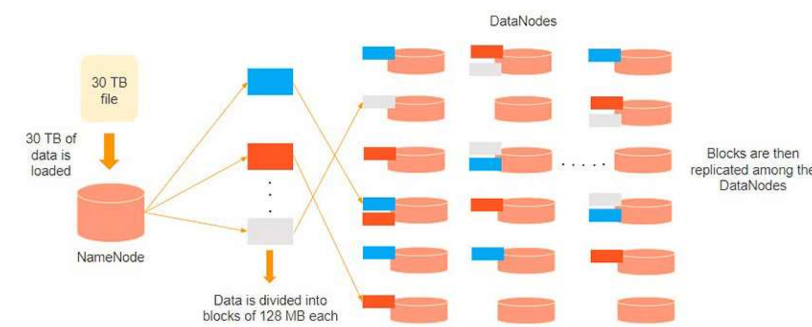
- Provides distributed storage
- Can be implemented on commodity hardware
- Provides data security
- Highly fault-tolerant - If one machine goes down, the data from that machine goes to the next machine

Master and Slave Nodes

- Master and slave nodes form the HDFS cluster. The name node is called the master, and the data nodes are called the slaves.
- The name node is responsible for the workings of the data nodes. It also stores the metadata.
- The data nodes read, write, process, and replicate the data. They also send signals, known as heartbeats, to the name node. These heartbeats show the status of the data node.
- Replication of the data is performed three times by default. It is done this way, so if a commodity machine fails, you can replace it with a new machine that has the same data.

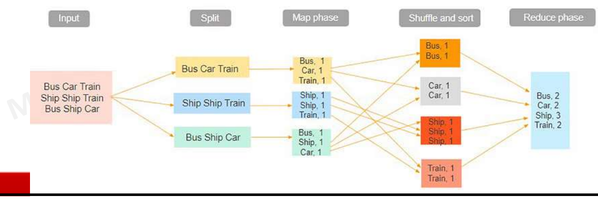


Example



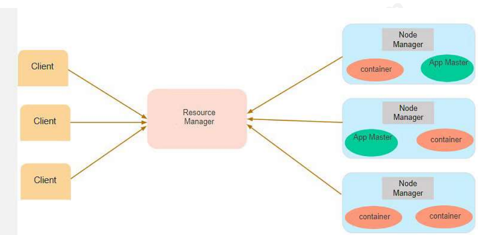
Hadoop MapReduce

- Hadoop MapReduce is the processing unit of Hadoop. In the MapReduce approach, the processing is done at the slave nodes, and the final result is sent to the master node.
- A data containing code is used to process the entire data. This coded data is usually very small in comparison to the data itself. You only need to send a few kilobytes worth of code to perform a heavy-duty process on computers.



Hadoop YARN

- Hadoop YARN stands for Yet Another Resource Negotiator. It is the resource management unit of Hadoop and is available as a component of Hadoop version 2.
- Hadoop YARN acts like an OS to Hadoop. It is a file system that is built on top of HDFS.
- It is responsible for managing cluster resources to make sure you don't overload one machine.
- It performs job scheduling to make sure that the jobs are scheduled in the right place



How Does Hadoop Work?

- The primary function of Hadoop is to process the data in an organised manner among the cluster of commodity software. The client should submit the data or program that needs to be processed. Hadoop HDFS stores the data. YARN, MapReduce divides the resources and assigns the tasks to the data. Let's know the working of Hadoop in detail.
  - The client input data is divided into 128 MB blocks by HDFS. Blocks are replicated according to the replication factor: various DataNodes house the unions and their duplicates.
  - The user can process the data once all blocks have been put on HDFS DataNodes.
  - The client sends Hadoop the MapReduce programme to process the data.
  - The user-submitted software was then scheduled by ResourceManager on particular cluster nodes.
  - The output is written back to the HDFS once processing has been completed by all nodes.

Hadoop Distributed File System

- HDFS is known as the Hadoop distributed file system. It is the allocated File System. It is the primary data storage system in Hadoop Applications. It is the storage system of Hadoop that is spread all over the system. In HDFS, the data is once written on the server, and it will continuously be used many times according to the need. The targets of HDFS are as follows.
  - The ability to recover from hardware failures in a timely manner
  - Access to Streaming Data
  - Accommodation of Large data sets
  - Portability
- Hadoop Distributed File System has two nodes included in it. They are the Name Node and Data Node.

Main components in Hadoop

- Name Node:
  - Name Node is the primary component of HDFS. Name Node maintains the file systems along with namespaces. Actual data can not be stored in the Name Node. The modified data, such as Metadata, block data etc., can be stored here.
- Data Node:
  - Data Node follows the instructions given by the Name Node. Data Nodes are also known as 'slave Nodes'. These nodes store the actual data provided by the client and simply follow the commands of the Name Node.
- Job Tracker:
  - The primary function of the Job Tracker is resource management. Job Tracker determines the location of the data by communicating with the Name Node. Job Tracker also helps in finding the Task Tracker. It also tracks the MapReduce from Local Node to Slave Node. In Hadoop, there is only one instance of Job Trackers. Job Tracker monitors the individual Task Tracker and tracks the status. Job Tracker also helps in the execution of MapReduce in Hadoop.
- Task Tracker:
  - Task Tracker is the slave daemon in the cluster which accepts all the instructions from the Job Tracker. Task Tracker runs on its process. The task trackers monitor all the tasks by capturing the input and output codes. The Task Tracker helps in mapping, shuffling and reducing the data operations. Task Tracker arranges different slots to perform different tasks. Task Tracker continuously updates the status of the Job Tracker. It also informs about the number of slots available in the cluster. In case the Task Tracker is unresponsive, then Job Tracker assigns the work to some other nodes.

How Hadoop Improves on Traditional Databases

- Understanding what is Hadoop requires further understanding on how it differs from traditional databases.
- Hadoop uses the HDFS (Hadoop Data File System) to divide the massive data amounts into manageable smaller pieces, then saved on clusters of community servers. This offers scalability and economy.
- Furthermore, Hadoop employs MapReduce to run parallel processing, which both stores and retrieves data faster than information residing on a traditional database. Traditional databases are great for handling predictable and constant workflows; otherwise, you need Hadoop's power of scalable infrastructure.

5 Advantages of Hadoop for Big Data

- Speed. Hadoop's concurrent processing, MapReduce model, and HDFS lets users run complex queries in just a few seconds.
- Diversity. Hadoop's HDFS can store different data formats, like structured, semi-structured, and unstructured.
- Cost-Effective. Hadoop is an open-source data framework.
- Resilient. Data stored in a node is replicated in other cluster nodes, ensuring fault tolerance.
- Scalable. Since Hadoop functions in a distributed environment, you can easily add more servers.

How Is Hadoop Being Used?

- Financial Sectors
  - Hadoop is used to detect fraud in the financial sector.
  - Hadoop is also used to analyze fraud patterns.
  - Credit card companies also use Hadoop to find out the exact customers for their products.
- Healthcare Sectors
  - Hadoop is used to analyze huge data such as medical devices, clinical data, medical reports etc.
  - Hadoop analyses and scans the reports thoroughly to reduce the manual work.
- Hadoop Applications in the Retail Industry
  - Retailers use Hadoop to improve their sales.
  - Hadoop also helped in tracking the products bought by the customers.
- Security and Law Enforcement
  - Data tools are used by the cops to chase criminals and predict their plans.
  - Hadoop is also used in defense, cybersecurity
- Hadoop Uses in Advertisements
  - Hadoop is used for capturing video, analyzing transactions and handling social media platforms.

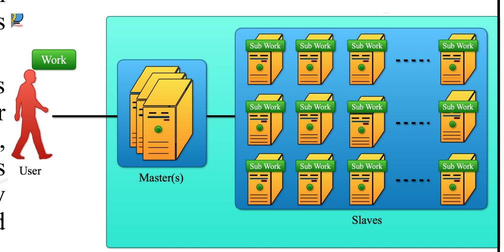
Challenges of Using Hadoop

- There's a steep learning curve. If you want to run a query in Hadoop's file system, you need to write MapReduce functions with Java, a process that is non-intuitive. Also, the ecosystem is made up of lots of components.
- Not every dataset can be handled the same. Hadoop doesn't give you a "one size fits all" advantage.
- MapReduce is limited. Yes, it's a great programming model, but MapReduce uses a file-intensive approach that isn't ideal for real-time interactive iterative tasks or data analytics.
- Security is an issue. There is a lot of data out there, and much of it is sensitive. Hadoop still needs to incorporate the proper authentication, data encryption, provisioning, and frequent auditing practices..

Distributed File System

What is HDFS

- HDFS is a distributed file system that provides access to data across Hadoop clusters.
- A cluster is a group of computers that work together. Like other Hadoop-related technologies, HDFS is a key tool that manages and supports analysis of very large volumes; petabytes and zettabytes of data.



Hadoop Cluster

Why HDFS? (Cost & Speed)

- Cost
  - HDFS is open-source software so that it can be used with zero licensing and support costs. It is designed to run on a regular computer.
- Speed
  - Large Hadoop clusters can read or write more than a terabyte of data per second. A cluster comprises multiple systems logically interconnected in the same network.
  - HDFS can easily deliver more than two gigabytes of data per second, per computer to MapReduce, which is a data processing framework of Hadoop.

Why HDFS? (Reliability)

- HDFS copies the data multiple times and distributes the copies to individual nodes. A node is a commodity server which is interconnected through a network device.
- HDFS then places at least one copy of data on a different server. In case, any of the data is deleted from any of the nodes; it can be found within the cluster.
- A regular file system, like a Linux file system, is different from HDFS with respect to the size of the data. In a regular file system, each block of data is small, usually about 51 bytes. However, in HDFS, each block is 128 Megabytes by default.
- A regular file system provides access to large data but may suffer from disk input/output problems mainly due to multiple seek operations.
- On the other hand, HDFS can read large quantities of data sequentially after a single seek operation. This makes HDFS unique since all of these operations are performed in a distributed mode.

Characteristics of HDFS

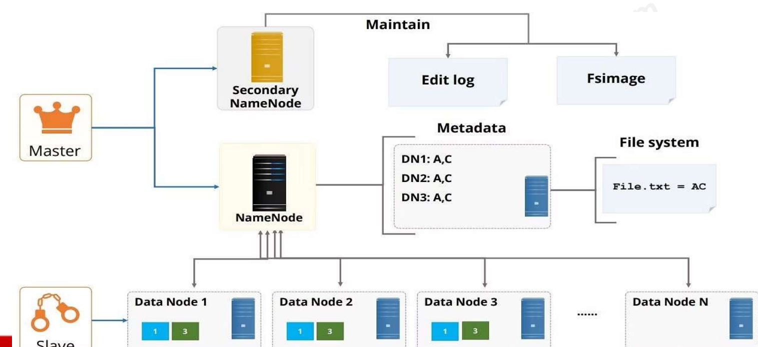
- HDFS has high fault-tolerance
- HDFS may consist of thousands of server machines. Each machine stores a part of the file system data. HDFS detects faults that can occur on any of the machines and recovers it quickly and automatically.
- HDFS has high throughput
- HDFS is designed to store and scan millions of rows of data and to count or add some subsets of the data. The time required in this process is dependent on the complexities involved.
- It has been designed to support large datasets in batch-style jobs. However, the emphasis is on high throughput of data access rather than low latency.
- HDFS is economical
- HDFS is designed in such a way that it can be built on commodity hardware and heterogeneous platforms, which is low-priced and easily available.

How Does HDFS Work

- HDFS stores files in several blocks. Data is divided into 128 Megabytes per block and replicated across local disks of cluster nodes.
- Each block is replicated to a few separate computers. The count of replication can be modified by the administrator.
- Metadata controls the physical location of a block and its replication within the cluster. It is stored in Name-Node. HDFS is the storage system for both input/output of MapReduce jobs.

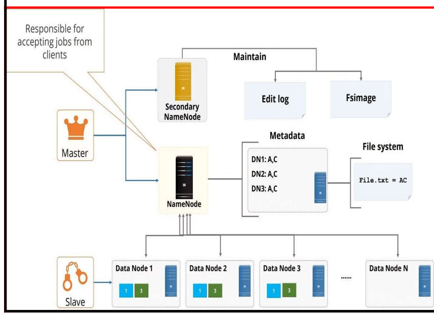
HDFS Architecture and Components

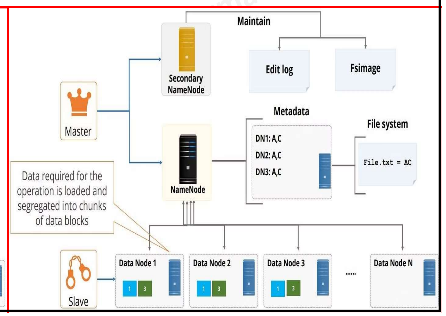
- HDFS architecture is known as the master and slave architecture which is shown below.



HDFS Architecture and Components

- A master node, that is the NameNode, is responsible for accepting jobs from the clients. Its task is to ensure that the data required for the operation is loaded and segregated into chunks of data blocks.

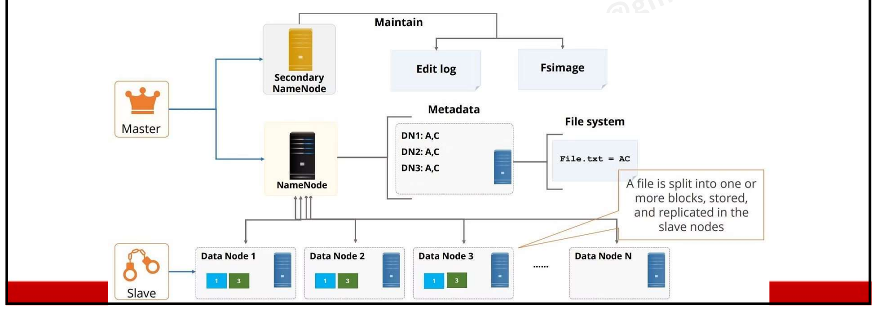




37

HDFS Architecture and Components

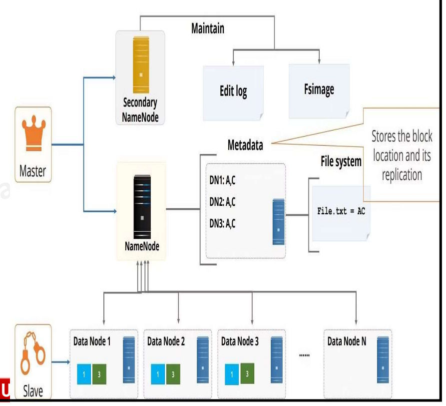
- HDFS exposes a file system namespace and allows user data to be stored in files. A file is split into one or more blocks, stored, and replicated in the slave nodes known as the DataNodes as shown in the section below.



38

HDFS Architecture and Components

- The data blocks are then distributed to the DataNode systems within the cluster. This ensures that the replicas of the data are maintained. DataNode serves to read or write requests. It also creates, deletes, and replicates blocks on the instructions from the NameNode.
- There is a Secondary NameNode which performs tasks for NameNode and is also considered as a master node. Prior to Hadoop 2.0.0, the NameNode was a Single Point of Failure, or SPOF, in an HDFS cluster.



39

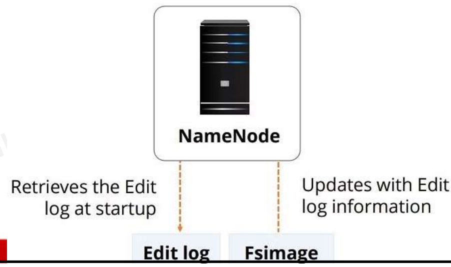
HDFS Components

- The main components of HDFS are:
  - Name-node
  - Secondary Name-node
  - File system
  - Metadata
  - Data-node

40

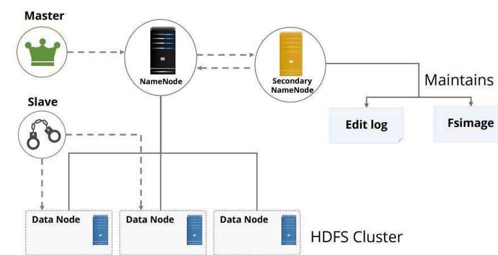
Name-node

- The NameNode server is the core component of an HDFS cluster. There can be only one NameNode server in an entire cluster. Namenode maintains and executes the file system namespace operation such as opening, closing, and renaming of files and directories, which are present in HDFS.



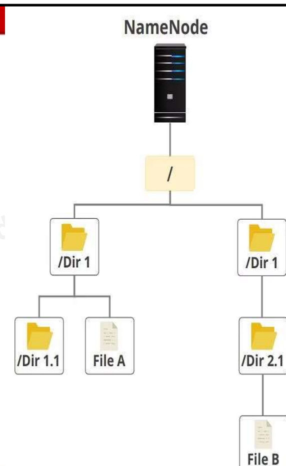
Secondary Name-node

- The Secondary NameNode server maintains the edit log and namespace image information in sync with the NameNode server. At times, the namespace images from the NameNode server are not updated; therefore, you cannot totally rely on the Secondary NameNode server for the recovery process.



File System

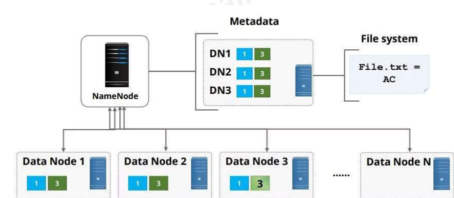
- HDFS exposes a file system namespace and allows user data to be stored in files. HDFS has a hierarchical file system with directories and files. The NameNode manages the file system namespace, allowing clients to work with files and directories.
- A file system supports operations like create, remove, move, and rename but not edit



BY MR. HUYNH NAM

Namenode: Operation

- NameNode maintains two persistent files; one a transaction log called an Edit Log and the other, a namespace image called a FsImage. The Edit Log records every change that occurs in the file system metadata such as creating a new file.
- The NameNode is a local filesystem that stores the Edit Log. The entire file system namespace including mapping of blocks, files, and file system properties is stored in FsImage. This is also stored in the NameNode local file system.

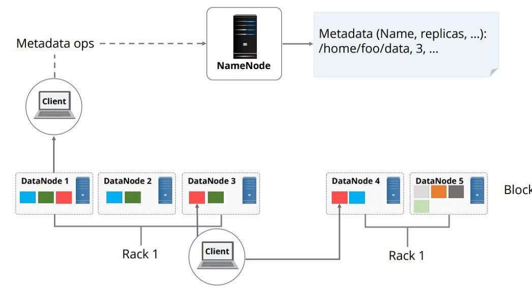


Metadata

- When new DataNodes join a cluster, metadata loads the blocks that reside on a specific DataNode into its memory at startup. Metadata then periodically loads the data at user-defined or default intervals.
- When the NameNode starts up, it retrieves the Edit Log and FsImage from its local file system. It then updates the FsImage with Edit Log information and stores a copy of the FsImage on the file system as a checkpoint.
- The metadata size is limited to the RAM available on the NameNode. A large number of small files would require more metadata than a small number of large files. Hence, the in-memory metadata management issue explains why HDFS favors a small number of large files.
- If a NameNode runs out of RAM, it will crash, and the applications will not be able to use HDFS until the NameNode is operational again.
- Data block split is an important process of HDFS architecture. As discussed earlier, each file is split into one or more blocks stored and replicated in DataNodes.

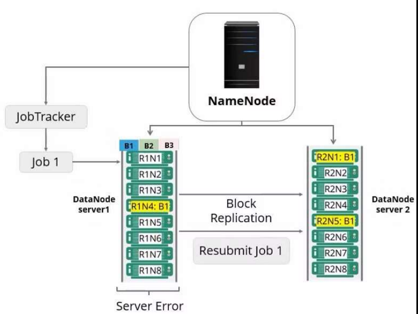
DataNode

- DataNodes manage names and locations of file blocks. By default, each file block is 128 Megabytes. However, this potentially reduces the amount of parallelism that can be achieved as the number of blocks per file decreases.



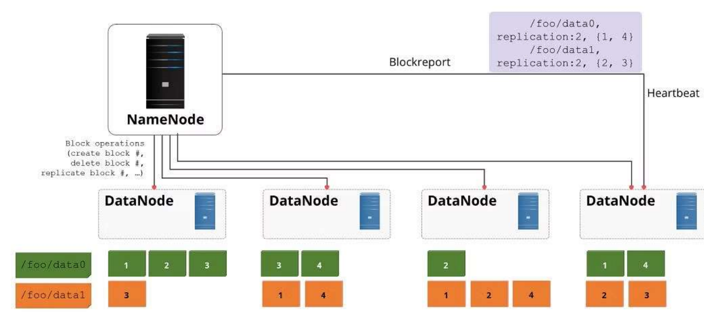
Block Replication Architecture

- Block replication refers to creating copies of a block in multiple data nodes. Usually, the data is split into the forms of parts such as part and part one.
- HDFS performs block replication on multiple data nodes so that if an error exists on one of the data nodes servers. The job tracker service resubmits the job to another data node server. The job tracker service is present in the name node server.



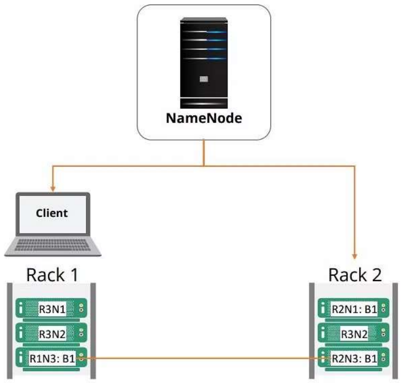
Replication Method

- In the replication method, each file is split into a sequence of blocks. All blocks except the last one in the file are of the same size. Blocks are replicated for fault tolerance.



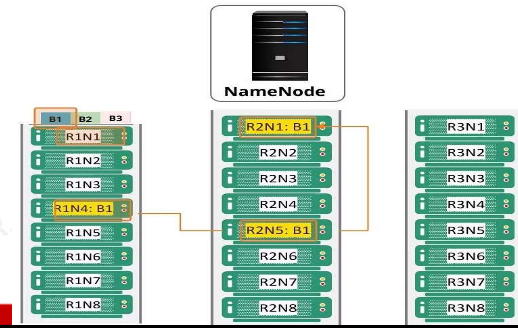
Data Replication Topology

- The topology of the replicas is critical to ensure the reliability of HDFS. Usually, each data is replicated thrice where the suggested replication topology is as follows.
- Place the first replica on the same node as that of the client. Place the second replica on a different rack from that of the first replica. Place the third replica on the same rack as that of the second one but on a different node.



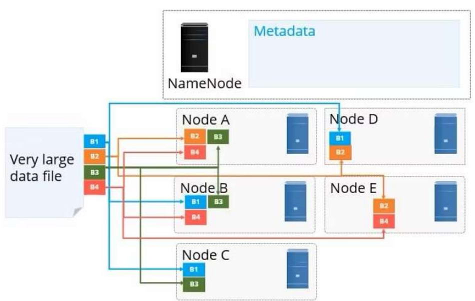
Data Replication Topology - Example

- The diagram illustrates a Hadoop cluster with three racks. A diagram for Replication and Rack Awareness in Hadoop is given below.

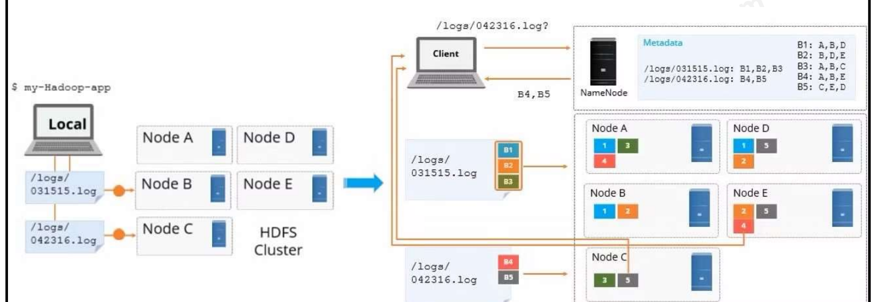


How Are Files Stored?

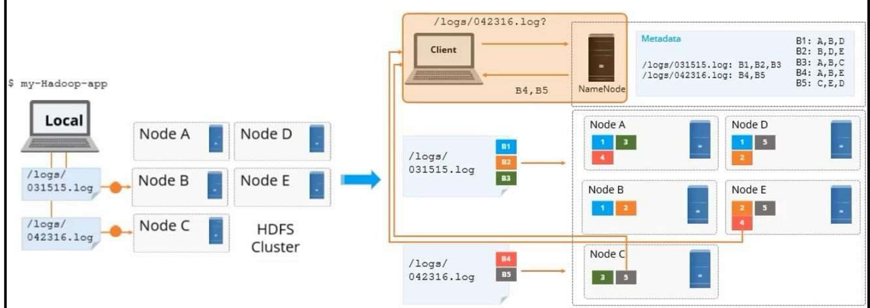
- Given context
  - we have a large data file that is divided into four blocks.
  - You might recall that the default size of each block is 128 megabytes.
  - The name node then carries metadata information of all blocks and its distribution.
  - Each block is replicated three times as shown in the diagram.



HDFS in Action - Example



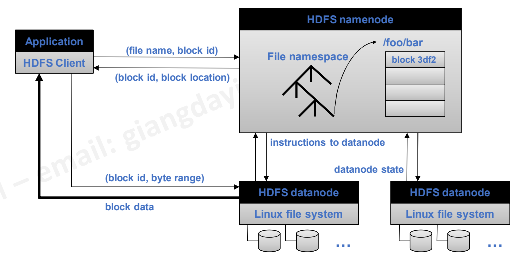
HDFS in Action - Example



53

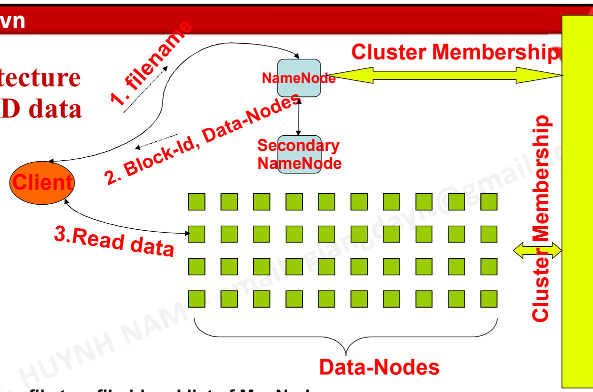
HDFS Working Flow

- Files stored as chunks
  - Fixed size (64MB)
- Reliability through replication
  - Each chunk replicated across 3+ chunk-servers
- Single master to coordinate access, keep metadata
  - Simple centralized management
- No data caching
  - Little benefit due to large datasets, streaming reads
- Simplify the API
  - Push some of the issues onto the client (e.g., data layout)



54

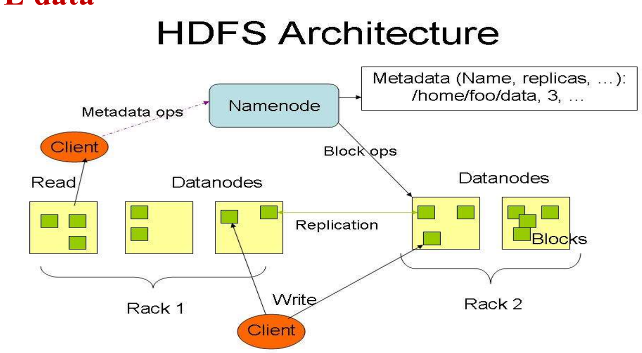
HDFS Architecture
How to READ data



NameNode : Maps a file to a file-id and list of MapNodes
DataNode : Maps a block-id to a physical location on disk
SecondaryNameNode: Periodic merge of Transaction log

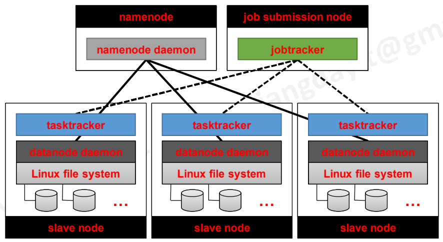
55

HDFS Architecture
How to WRITE data



56

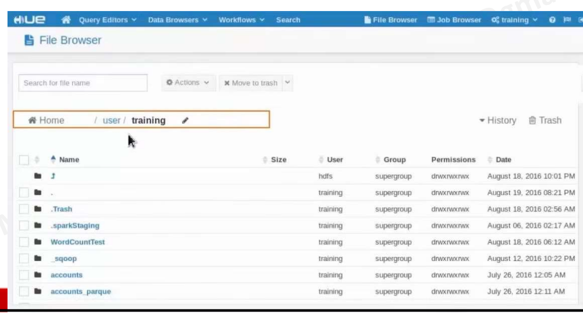
Putting everything together…



HDFS Command Line

Hue File Browser

- The file browser in Hue lets you view and manage your HDFS directories and files. Additionally, you can create, move, rename, modify, upload, download, and delete directories and files. You can also view the file contents.



Map Reduce

MR. HUYNH NAM

MapReduce Overview

- MapReduce is the process of making a list of objects and running an operation over each object in the list (i.e., map) to either produce a new list or calculate a single value (i.e., reduce).
- MapReduce is the processing engine of Hadoop that processes and computes large volumes of data. It is one of the most common engines used by Data Engineers to process Big Data. It allows businesses and other organizations to run calculations.
- Before MapReduce, these calculations were complicated. Now, programmers can tackle problems like these with relative ease. Data scientists have coded complex algorithms into frameworks so that programmers can use them.
- MapReduce runs across a network of low-cost commodity machines. There are two phases in the MapReduce programming model:
  - Mapping
  - Reducing

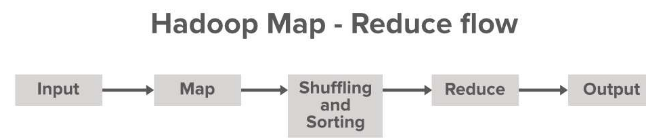
Map Reduce generic workflow

- Iterate over a large number of records
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output

Map

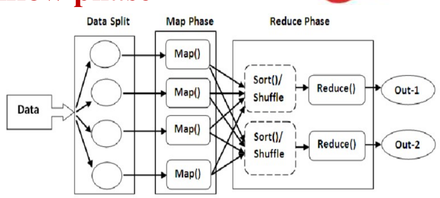
Reduce

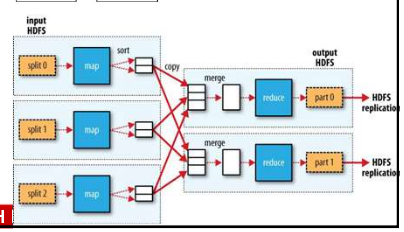
Key idea: provide a functional abstraction for these two operations



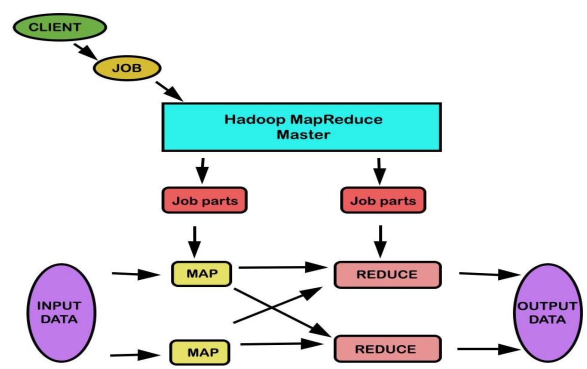
Mapping and Reducing workflow phase

- Input Data
  - Hadoop accepts data in various formats and stores it in HDFS. This input data is worked upon by multiple map tasks.
- Map Tasks
  - Map reads the data, processes it, and generates key-value pairs. The number of map tasks depends upon the input file and its format.
  - Typically, a file in a Hadoop cluster is broken down into blocks, each with a default size of 128 MB. Depending upon the size, the input file is split into multiple chunks. A map task then runs for each chunk. The mapper class has mapper functions that decide what operation is to be performed on each chunk.
- Reduce Tasks
  - In the reducing phase, a reducer class performs operations on the data generated from the map tasks through a reducer function. It shuffles, sorts, and aggregates the intermediate key-value pairs (tuples) into a set of smaller tuples.
- Output
  - The smaller set of tuples is the final output and gets stored in HDFS.





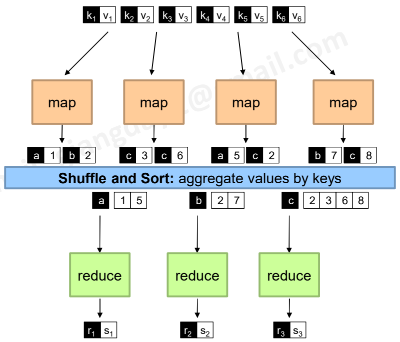
MapReduce Architecture



www.flex.edu.vn

MapReduce in action

- Programmers specify two functions:
  - **map** $(k_1, v_1) \rightarrow [<k_2, v_2>]$
  - **reduce** $(k_2, [v_2]) \rightarrow [<k_3, v_3>]$
  - All values with the same key are sent to the same reducer



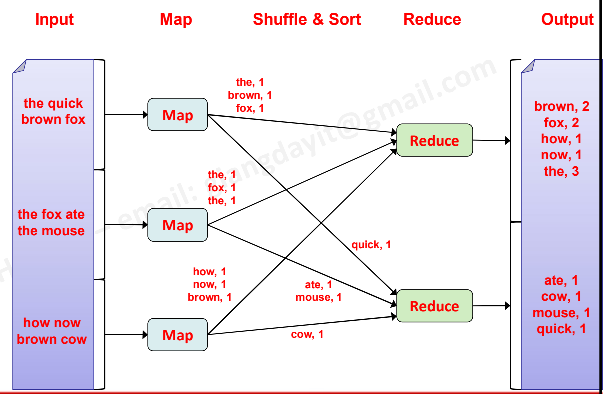
BY MR. HUYNH NAM

65

www.flex.edu.vn

original

- Cho các file1.data, file2.data, file3.data chứa lần lượt dữ liệu
- File1.data:
  - the quick
  - brown fox
- File2.data
  - the fox ate
  - the mouse
- File3.data
  - how now
  - brown cow
- Hãy đếm số từ trong các file trên trong hệ thống Hadoop với Map-Reduce



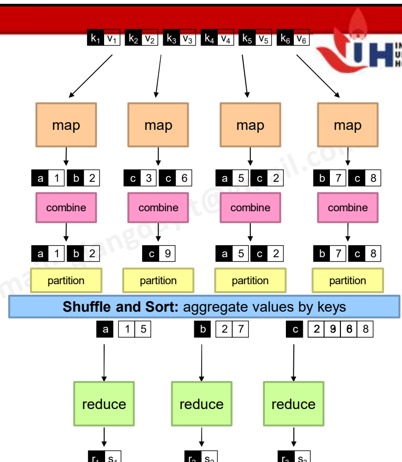
BY MR. HUYNH NAM

66

www.flex.edu.vn

Map Reduce in action

- Not quite…usually, programmers also specify:
  - **combine** $(k', v') \rightarrow <k', v'>^*$
    - Mini-reducers that run in memory after the map phase
    - Used as an optimization to reduce network traffic
  - **partition** (k', number of partitions) → partition for k'
    - Often a simple hash of the key, e.g., hash(k') mod n
    - Divides up key space for parallel reduce operations

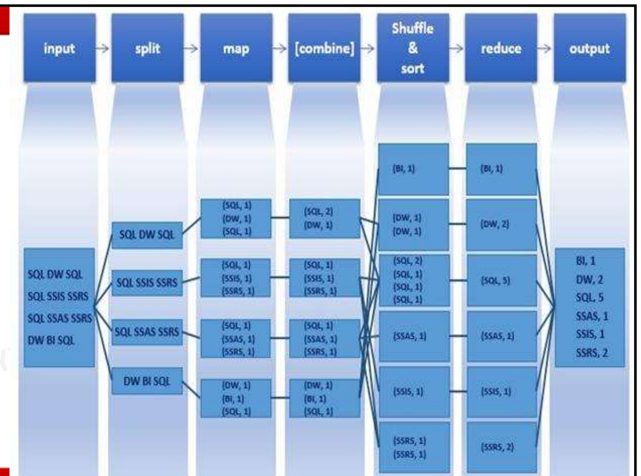


BY MR. HUYNH NAM

67

www.flex.edu.vn

Combiner example

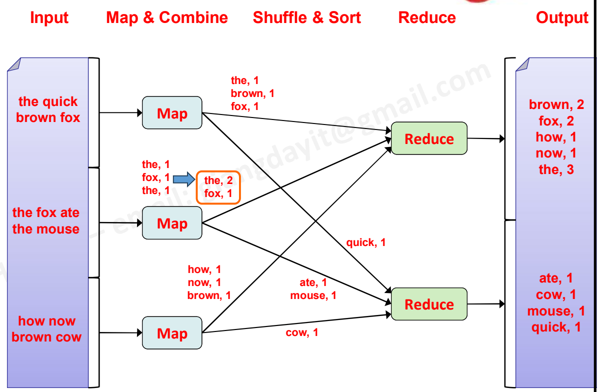


68

www.flex.edu.vn

combiner

- Cho các file1.data, file2.data, file3.data chứa lần lượt dữ liệu
- File1.data:
  - the quick
  - brown fox
- File2.data
  - the fox ate
  - the mouse
- File3.data
  - how now
  - brown cow
- Hãy đếm số từ trong các file trên trong hệ thống Hadoop với Map-Reduce

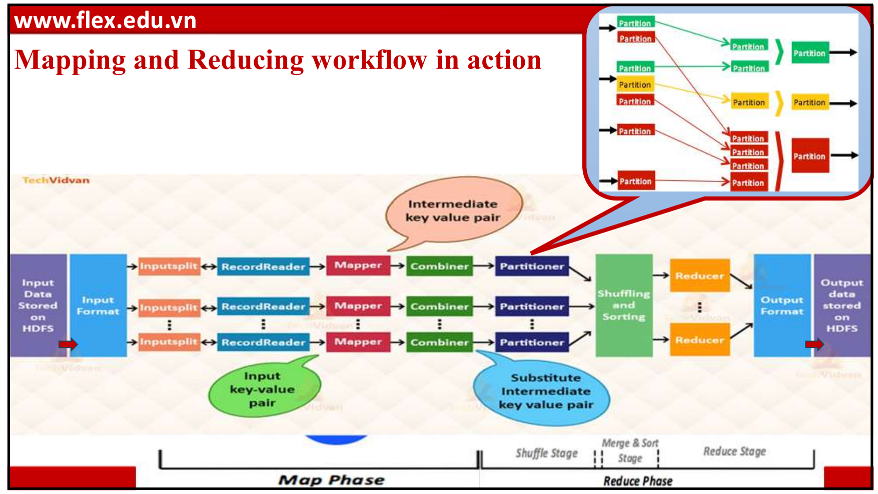


BY MR. HUYNH NAM

69

www.flex.edu.vn

Mapping and Reducing workflow in action

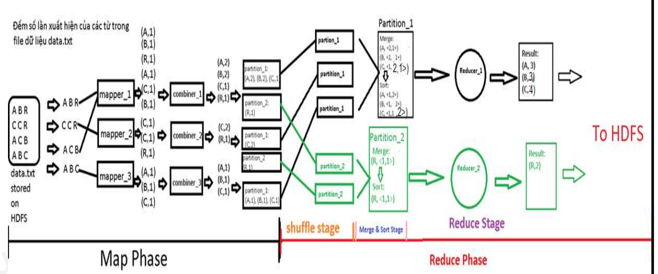


BY MR. HUYNH NAM

70

www.flex.edu.vn

- Vẽ sơ đồ Map-reduce (MR) để thực hiện đếm số lượng ký tự của các chữ cái trong bảng Alphabet ([a-Z], không phân biệt hoa thường) xuất hiện trong các tập tin
- Input data.txt
  - A B R
  - C C R
  - A C B
  - A B C
- Giả định rằng:
  - Dữ liệu trên dòng [1,3] được đưa vào mapper_1, dòng [2] được đưa vào mapper_2 và dòng [4] được đưa vào mapper_3
  - Key A, B, C được gắn nhãn Partition 1, còn R được gắn nhãn Partition 2
- Output
  - A: 3
  - B: 3
  - C: 4
  - R: 2



BY MR. HUYNH NAM

71

www.flex.edu.vn

Exercise

- Cho hệ thống Hadoop lưu trữ thông tin về lương của ~200,000 nhân viên trong 10 năm có cấu trúc như sau:

Số thứ tự	Mã nhân viên	Tên nhân viên	Lương (USD)
1	1	An	250
2	2	Bình	100
3	3	Duy	200
4	4	Duy	500
5	4	Duy	1000
6	3	Duy	250
7	1	An	210
8	4	Duy	170
9	2	Bình	1500
10	1	An	200
11	3	Duy	450
12	3	Duy	650

- Hãy thống kê lương nhỏ nhất của các nhân viên. Biết rằng hệ thống có các partitions tham gia. Hệ thống file tự động ánh xạ các data block thành các Input Split theo số thứ tự như sau: Input Split 1: [1,2,3]; Input Split 2: [4,5,6,7]; Input Split 3: [8,9]; Input Split 4: [10,11,12]. Input Formatter được mô tả như sau: dữ liệu lưu trữ trên từng dòng và các thành phần được phân cách bởi dấu ‘,’.Các combiner thực hiện tương ứng với từng mapper. Ngoài ra, tổng dung lượng lưu trữ dữ liệu là 2 TB và hệ thống lưu trữ là: HDFS_salary.
- Hãy vẽ sơ đồ hoạt động của Map Reduce trên hệ thống với môi trường có tham gia bởi các Combiners và Partitioners. Biết rằng hàm ánh xạ các Partitioners được xây dựng bởi hash function:

```
function par(employee_id){
        return  employee_id % 3
}
```

- Trong đó: employee_id: mã nhân viên

BY MR. HUYNH NAM

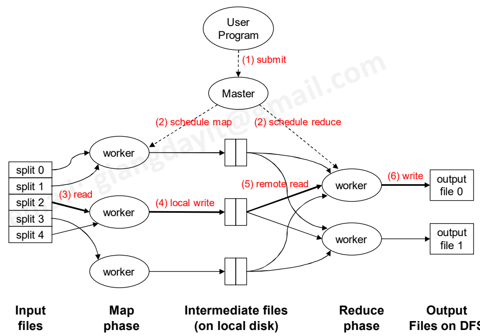
72

MapReduce "Runtime"

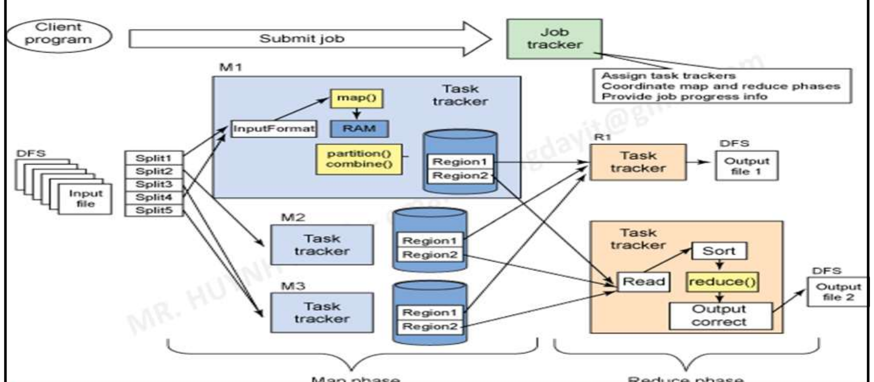
- Handles scheduling
  - Assigns workers to map and reduce tasks
- Handles "data distribution"
  - Moves processes to data
- Handles synchronization
  - Gathers, sorts, and shuffles intermediate data
- Handles errors and faults
  - Detects worker failures and restarts
- Everything happens on top of a distributed FS

MapReduce Implementations

- Google has a proprietary implementation in C++
  - Bindings in Java, Python
- Hadoop is an open-source implementation in Java
  - Development led by Yahoo, now an Apache project
  - Used in production at Yahoo, Facebook, Twitter, LinkedIn, Netflix, …
  - The *de facto* big data processing platform
  - Large and expanding software ecosystem
- Lots of custom research implementations
  - For GPUs, cell processors, etc.



Behind in the Sense



Word Count implementation in Python

python/MapReduce/HadoopStreaming/mapper.py

```
#!/usr/bin/env python

import sys

# Read each line from stdin

for line in sys.stdin:

  # Get the words in each line
  words = line.split()

  # Generate the count for each word
  for word in words:

    # Write the key-value pair to stdout to be processed by
    # the reducer.
    # The key is anything before the first tab character and the
    #value is anything after the first tab character.
    print '{0}\t{1}'.format(word, 1)
```

python/MapReduce/HadoopStreaming/reducer.py

```
#!/usr/bin/env python

import sys

curr_word = None
curr_count = 0

# Process each key-value pair from the mapper
for line in sys.stdin:

  # Get the key and value from the current line
  word, count = line.split('\t')

  # Convert the count to an int
  count = int(count)

  # If the current word is the same as the previous word,
  # increment its count, otherwise print the words count
  # to stdout
  if word == curr_word:
     curr_count += count
  else:

     # Write word and its number of occurrences as a key-value
     # pair to stdout
     if curr_word:
        print '{0}\t{1}'.format(curr_word, curr_count)

     curr_word = word

        curr_count = count

 # Output the count for the last word
 if curr_word == word:
   print '{0}\t{1}'.format(curr_word, curr_count)
```

YARN

YARN is the acronym for Yet Another Resource Negotiator. YARN is a resource manager created by separating the processing engine and the management function of MapReduce. It monitors and manages workloads, maintains a multi-tenant environment, manages the high availability features of Hadoop, and implements security controls.

Table of Contents

What is Yarn?

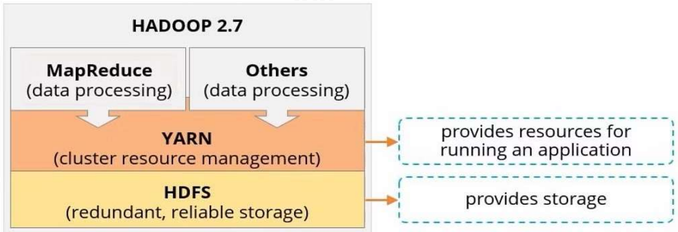
- Before 2012, users could write MapReduce programs using scripting languages such as Java, Python, and Ruby. They could also use Pig, a language used to transform data. No matter what language was used, its implementation depended on the MapReduce processing model.
- In May 2012, during the release of Hadoop version 2.0, YARN was introduced. You are no longer limited to working with the MapReduce framework anymore as YARN supports multiple processing models in addition to MapReduce, such as Spark. Other features of YARN include significant performance improvement and a flexible execution engine.

YARN - Use Case

- Yahoo was the first company to embrace Hadoop and this became a trendsetter within the Hadoop ecosystem. In late 2012, Yahoo struggled to handle iterative and stream processing of data on the Hadoop infrastructure due to MapReduce limitations.
- Both iterative and stream processing was important for Yahoo in facilitating its move from batch computing to continuous computing.
- After implementing YARN in the first quarter of 2013, Yahoo installed more than 30,000 production nodes on
  - Spark for iterative processing
  - Storm for stream processing
  - Hadoop for batch processing, allowing it to handle more than 100 billion events such as clicks, impressions, email content, metadata, and so on per day.
- This was possible only after YARN was introduced and multiple processing frameworks were implemented. The single-cluster approach provides a number of advantages, including:
  - Higher cluster utilization, where resources unutilized by a framework can be consumed by another
  - Lower operational costs because only one "do-it-all" cluster needs to be managed
  - Reduced data motion as there's no need to move data between Hadoop YARN and systems running on different clusters of computers

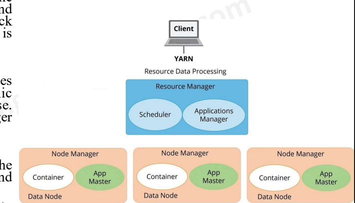
YARN Infrastructure

- The YARN Infrastructure is responsible for providing computational resources such as CPUs or memory needed for application executions.
- YARN infrastructure and HDFS are completely independent. The former provides resources for running an application while the latter provides storage.
- The MapReduce framework is only one of the many possible frameworks that run on YARN. The fundamental idea of MapReduce version-2 is to split the two major functionalities of resource management and job scheduling and monitoring into separate daemons.



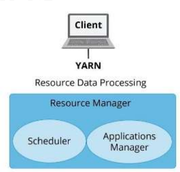
YARN and its Architecture

- The three important elements of the YARN architecture are:
  - Resource Manager
    - The ResourceManager, or RM, which is usually one per cluster, is the master server. Resource Manager knows the location of the DataNode and how many resources they have. This information is referred to as Rack Awareness. The RM runs several services, the most important of which is the Resource Scheduler that decides how to assign the resources.
  - Application Master
    - The Application Master is a framework-specific process that negotiates resources for a single application, that is, a single job or a directed acyclic graph of jobs, which runs in the first container allocated for the purpose. Each Application Master requests resources from the Resource Manager and then works with containers provided by Node Managers.
  - Node Managers
    - The Node Managers can be many in one cluster. They are the slaves of the infrastructure. When it starts, it announces itself to the RM and periodically sends a heartbeat to the RM.
    - Each Node Manager offers resources to the cluster. The resource capacity is the amount of memory and the number of v-cores, short for the virtual core. At run-time, the Resource Scheduler decides how to use this capacity. A container is a fraction of the NodeManager capacity, and it is used by the client to run a program. Each Node Manager takes instructions from the ResourceManager and reports and handles containers on a single node.



YARN Architecture Element - Resource Manager

- The first element of YARN architecture is ResourceManager. The RM mediates the available resources in the cluster among competing applications with the goal of maximum cluster utilization.
- It includes a pluggable scheduler called the YarnScheduler, which allows different policies for managing constraints such as capacity, fairness, and Service Level Agreements. The Resource Manager has two main components - Scheduler and Applications Manager. Let us understand each of them in detail.
- Resource Manager Component – Scheduler
  - The Scheduler is responsible for allocating resources to various running applications depending on the common constraints of capacities, queues, and so on. The Scheduler does not monitor or track the status of the application. Also, it does not restart the tasks in case of any application or hardware failures. The Scheduler performs its function based on the resource requirements of the applications. It does so based on the abstract notion of a resource container that incorporates elements such as memory, CPU, disk, and network. The Scheduler has a policy plugin which is responsible for partitioning the cluster resources among various queues and applications. The current MapReduce schedulers such as the Capacity Scheduler and the Fair Scheduler are some examples of the plug-in.
  - The Capacity Scheduler supports hierarchical queues to enable a more predictable sharing of cluster resources.
- Resource Manager Component - Application Manager
  - The Application Manager is an interface which maintains a list of applications that have been submitted, currently running, or completed. The Application Manager is responsible for accepting job-submissions, negotiating the first container for executing the application-specific Application Master and restarting the Application Master container on failure.

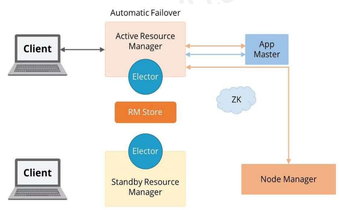


How Does the Resource Manager Operate?

- The Resource Manager communicates with the clients through an interface called the Client Service. A client can submit or terminate an application and gain information about the scheduling queue or cluster statistics through the Client Service.
- Administrative requests are served by a separate interface called the Admin Service through which operators can get updated information about the cluster operation.
- In parallel, the Resource Tracker Service receives node heartbeats from the Node Manager to track new or decommissioned nodes.
- The NM Liveliness Monitor and Nodes List Manager keep an updated status of which nodes are healthy so that the Scheduler and the Resource Tracker Service can allocate work appropriately.
- The Application Master Service manages Application Masters on all nodes, keeping the Scheduler informed. The AM Liveliness Monitor keeps a list of Application Masters and their last heartbeat times to let the Resource Manager know what applications are healthy on the cluster.
- Any Application Master that does not send a heartbeat within a certain interval is marked as dead and re-scheduled to run on a new container.

Resource Manager in High Availability Mode

- Before Hadoop 2.4, the Resource Manager was the single point of failure in a YARN cluster. The High Availability, or HA, the feature adds redundancy in the form of an Active/Standby Resource Manager pair to remove this single point of failure.
- Resource Manager HA is realized through the Active/Standby architecture. At any point in time, one of the RMs is active and one or more RMs are in Standby mode waiting to take over, should anything happen to the Active. The trigger to transition-to-active comes from either the admin through the Command-Line Interface or through the integrated failover-controller.
- The RMs have an option to invade the zookeeper base active standby Elector to decide which RMs should be active. Only active go down or become unresponsive, another RMs is automatically Elector to be active. Note there is no need to run a separate ZKFC Demon like in HDFS. Because the active standby Elector embedded in RMs acts as a failure to a detector and leads to an Elector.



YARN Architecture Element - Application Master

- The second element of YARN architecture is the Application Master. The Application Master in YARN is a framework-specific library, which negotiates resources from the RM and works with the NodeManager or Managers to execute and monitor containers and their resource consumption.
- While an application is running, the Application Master manages the application lifecycle, dynamic adjustments to resource consumption, execution flow, faults, and it provides status and metrics.
- The Application Master is architected to support a specific framework and can be written in any language. It uses extensible communication protocols with the Resource Manager and the Node Manager.
- The Application Master can be customized to extend the framework or run any other code. Because of this, the Application Master is not considered trustworthy and is not run as a trusted service.
- In reality, every application has its own instance of an Application Master. However, it is feasible to implement an Application Master to manage a set of applications, for example, an Application Master for Pig or Hive to manage a set of MapReduce jobs.

YARN Architecture Element - Node Manager

- The third element of YARN architecture is the Node Manager. When a container is leased to an application, the NodeManager sets up the container environment. The environment includes the resource constraints specified in the lease and any kind of dependencies, such as data or executable files.
- The Node Manager monitors the health of the node, reporting to the ResourceManager when a hardware or software issue occurs so that the Scheduler can divert resource allocations to healthy nodes until the issue is resolved. The Node Manager also offers a number of services to containers running on the node such as a log aggregation service.
- The Node Manager runs on each node and manages the activities such as container lifecycle management, container dependencies, container leases, node and container resource usage, node health, and log management and reports node and container status to the Resource Manager.

Node Manager Component: YARN Container

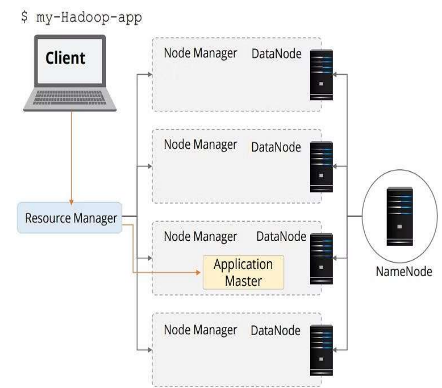
- A YARN container is a collection of a specific set of resources to use in certain amounts on a specific node. It is allocated by the ResourceManager on the basis of the application. The Application Master presents the container to the Node Manager on the node where the container has been allocated, thereby gaining access to the resources.
- The Application Master must provide a Container Launch Context or CLC. This includes information such as Environment variables, dependencies on the requirement of data files or shared objects prior to the launch, security tokens, and the command to create the process to launch the application.
- The CLC supports the Application Master to use containers. This helps to run a variety of different kinds of work, from simple shell scripts to applications to a virtual operating system.

Applications on YARN

- Owing to YARN is the generic approach, a Hadoop YARN cluster runs various work-loads. This means a single Hadoop cluster in your data center can run MapReduce, Storm, Spark, Impala, and more.
- Running an Application through YARN. Broadly, there are five steps involved in YARN to run an application:
  - The client submits an application to the Resource Manager
  - The ResourceManager allocates a container
  - The Application Master contacts the related Node Manager
  - The Node Manager launches the container
  - The container executes the Application Master

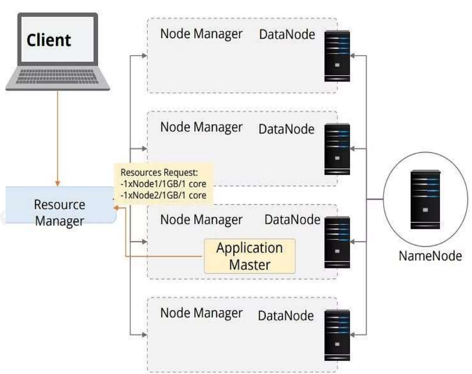
Step 1 - Application submitted to the Resource Manager

- Users submit applications to the Resource Manager by typing the Hadoop jar command.
- The Resource Manager maintains the list of applications on the cluster and available resources on the Node Manager. The Resource Manager determines the next application that receives a portion of the cluster resource. The decision is subject to many constraints such as queue capacity, Access Control Lists, and fairness.



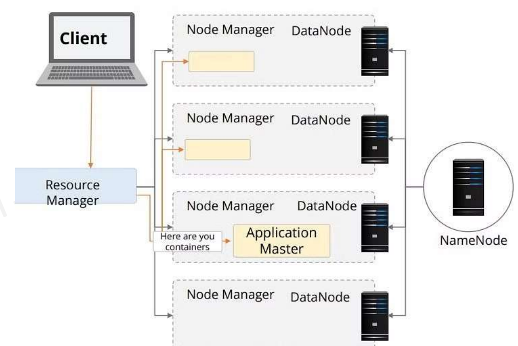
Step 2 - Resource Manager allocates Container

- When the Resource Manager accepts a new application submission, one of the first decisions the Scheduler makes is selecting a container. Then, the Application Master is started and is responsible for the entire life-cycle of that particular application.
- First, it sends resource requests to the ResourceManager to ask for containers to run the application's tasks.
- A resource request is simply a request for a number of containers that satisfy resource requirements such as the following:
  - Amount of resources expressed as megabytes of memory and CPU shares Preferred location, specified by hostname or rackname, Priority within this application and not across multiple applications.
  - The Resource Manager allocates a container by providing a container ID and a hostname, which satisfies the requirements of the Application Master.



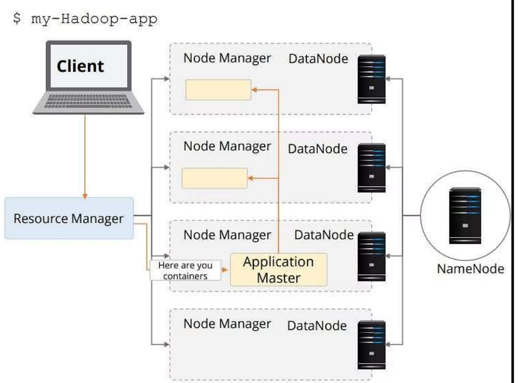
Step 3 - Application Master contacts Node Manager

- After a container is allocated, the Application Master asks the Node Manager managing the host on which the container was allocated to use these resources to launch an application-specific task. This task can be any process written in any framework, such as a MapReduce task.



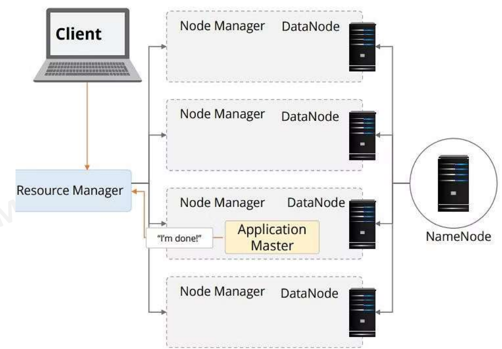
Step 4 -Resource Manager Launches Container

- The NodeManager does not monitor tasks; it only monitors the resource usage in the containers.
- For example, it kills a container if it consumes more memory than initially allocated.
- Throughout its life, the Application Master negotiates containers to launch all of the tasks needed to complete its application. It also monitors the progress of an application and its tasks, restarts failed tasks in newly requested containers, and reports progress back to the client that submitted the application.



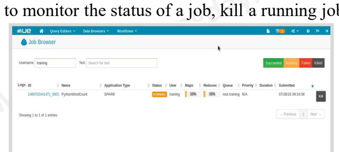
Step 5 - Container Executes the Application Master

- After the application is complete, the Application Master shuts itself and releases its own container. Though the ResourceManager does not monitor the tasks within an application, it checks the health of the ApplicationMaster. If the ApplicationMaster fails, it can be restarted by the ResourceManager in a new container. Thus, the resource manager looks after the ApplicationMaster, while the ApplicationMaster looks after the tasks.



Tools for YARN Development

- Hadoop includes three tools for YARN developers:
  - YARN Web UI
    - YARN web UI runs on 8088 port by default. It also provides a better view than Hue; however, you cannot control or configure from YARN web UI.
  - Hue Job Browser
    - The Hue Job Browser allows you to monitor the status of a job, kill a running job, and view logs.



  - YARN Command Line
    - Most of the YARN commands are for the administrator rather than the developer.
- These tools enable developers to submit, monitor, and manage jobs on the YARN cluster.

YARN commands

- To list all commands of YARN:

```
-yarn -help
```

It lists all the commands of yarn.

- To print the version:

```
- yarn -version
```

It prints the version.

- To view logs of a specified application ID:

```
- yarn logs -applicationId <app-id>
```

Hadoop Eco-System

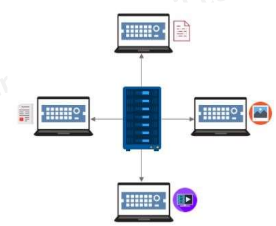
Table of Contents

Hadoop Eco-System

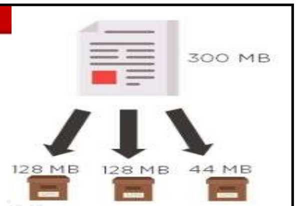
- Hadoop is a framework that manages big data storage by means of parallel and distributed processing. Hadoop is comprised of various tools and frameworks that are dedicated to different sections of data management, like storing, processing, and analyzing. The Hadoop ecosystem covers Hadoop itself and various other related big data tools.

HDFS

- In the traditional approach, all data was stored in a single central database. With the rise of big data, a single database was not enough to handle the task. The solution was to use a distributed approach to store the massive volume of information. Data was divided up and allocated to many individual databases. HDFS is a specially designed file system for storing huge datasets in commodity hardware, storing information in different formats on various machines.



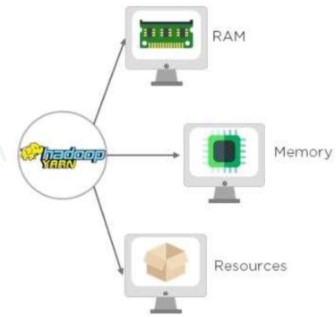
Components in HDFS



- There are two components in HDFS:
  - NameNode - NameNode is the master daemon. There is only one active NameNode. It manages the DataNodes and stores all the metadata.
  - DataNode - DataNode is the slave daemon. There can be multiple DataNodes. It stores the actual data.
- So, we spoke of HDFS storing data in a distributed fashion, but did you know that the storage system has certain specifications? HDFS splits the data into multiple blocks, defaulting to a maximum of 128 MB. The default block size can be changed depending on the processing speed and the data distribution.
- As seen from the above image, we have 300 MB of data. This is broken down into 128 MB, 128 MB, and 44 MB. The final block handles the remaining needed storage space, so it doesn't have to be sized at 128 MB. This is how data gets stored in a distributed manner in HDFS.
- Now that you have an overview of HDFS, it is also vital for you to understand what it sits on and how the HDFS cluster is managed. That is done by YARN, and that's what we're looking at next.

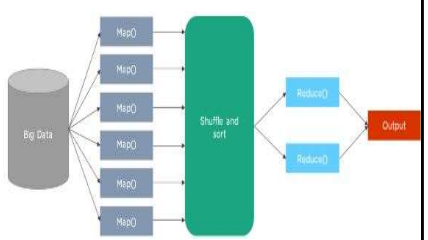
YARN (Yet Another Resource Negotiator)

- YARN is an acronym for Yet Another Resource Negotiator. It handles the cluster of nodes and acts as Hadoop's resource management unit. YARN allocates RAM, memory, and other resources to different applications.
- YARN has two components :
  - ResourceManager (Master) - This is the master daemon. It manages the assignment of resources such as CPU, memory, and network bandwidth.
  - NodeManager (Slave) - This is the slave daemon, and it reports the resource usage to the Resource Manager.



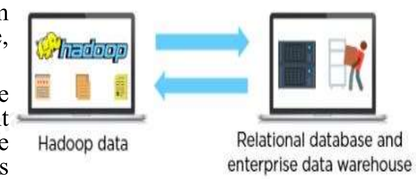
MapReduce

- Hadoop data processing is built on MapReduce, which processes large volumes of data in a parallelly distributed manner.
- As we see, we have our big data that needs to be processed, with the intent of eventually arriving at an output. So in the beginning, input data is divided up to form the input splits. The first phase is the Map phase, where data in each split is passed to produce output values. In the shuffle and sort phase, the mapping phase's output is taken and grouped into blocks of similar data. Finally, the output values from the shuffling phase are aggregated. It then returns a single output value.

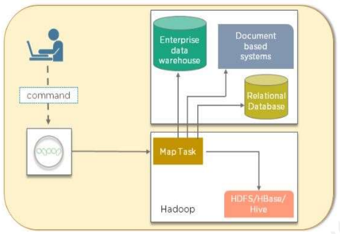


Sqoop

- Sqoop is used to transfer data between Hadoop and external datastores such as relational databases and enterprise data warehouses. It imports data from external datastores into HDFS, Hive, and HBase.
- As seen below, the client machine gathers code, which will then be sent to Sqoop. The Sqoop then goes to the Task Manager, which in turn connects to the enterprise data warehouse, documents-based systems, and RDBMS. It can map those tasks into Hadoop.

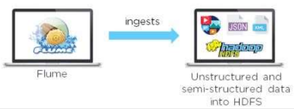


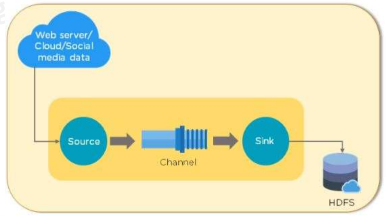
Flume



As you can see below, data is taken from various sources, depending on your organization's needs. It then goes through the source, channel, and sink. The sink feature ensures that everything is in sync with the requirements. Finally, the data is dumped into HDFS.

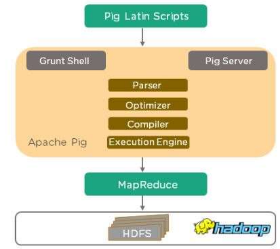
Flume is another data collection and ingestion tool, a distributed service for collecting, aggregating, and moving large amounts of log data. It ingests online streaming data from social media, logs files, web server into HDFS.





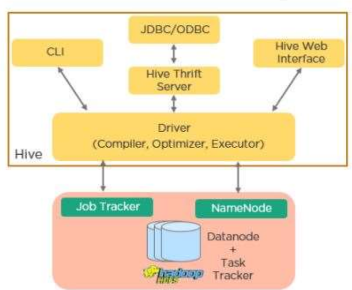
Pig

- Apache Pig was developed by Yahoo researchers, targeted mainly towards non-programmers. It was designed with the ability to analyze and process large datasets without using complex Java codes. It provides a high-level data processing language that can perform numerous operations without getting bogged down with too many technical concepts.
- It consists of:
  - Pig Latin - This is the language for scripting
  - Pig Latin Compiler - This converts Pig Latin code into executable code
- Pig also provides Extract, Transfer, and Load (ETL), and a platform for building data flow. Did you know that ten lines of Pig Latin script equals approximately 200 lines of MapReduce job? Pig uses simple, time-efficient steps to analyze datasets.



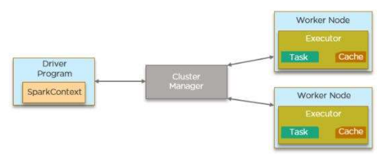
Hive

- Hive uses SQL (Structured Query Language) to facilitate the reading, writing, and management of large datasets residing in distributed storage. The hive was developed with a vision of incorporating the concepts of tables and columns with SQL since users were comfortable with writing queries in SQL.
- Apache Hive has two major components:
  - Hive Command Line
  - JDBC/ ODBC driver
- The Java Database Connectivity (JDBC) application is connected through JDBC Driver, and the Open Database Connectivity (ODBC) application is connected through ODBC Driver. Commands are executed directly in CLI. Hive driver is responsible for all the queries submitted, performing the three steps of compilation, optimization, and execution internally. It then uses the MapReduce framework to process queries.
- Hive's architecture is shown below



Spark

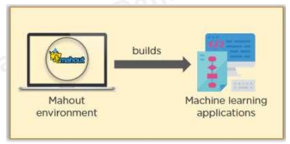
- Spark is a huge framework in and of itself, an open-source distributed computing engine for processing and analyzing vast volumes of real-time data. It runs 100 times faster than MapReduce. Spark provides an in-memory computation of data, used to process and analyze real-time streaming data such as stock market and banking data, among other things.



- As seen from the above image, the MasterNode has a driver program. The Spark code behaves as a driver program and creates a SparkContext, which is a gateway to all of the Spark functionalities. Spark applications run as independent sets of processes on a cluster. The driver program and Spark context take care of the job execution within the cluster. A job is split into multiple tasks that are distributed over the worker node. When an RDD is created in the Spark context, it can be distributed across various nodes. Worker nodes are slaves that run different tasks. The Executor is responsible for the execution of these tasks. Worker nodes execute the tasks assigned by the Cluster Manager and return the results to the SparkContext.

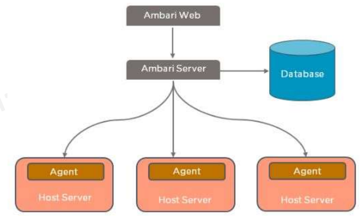
Mahout

- Mahout is used to create scalable and distributed machine learning algorithms such as clustering, linear regression, classification, and so on. It has a library that contains built-in algorithms for collaborative filtering, classification, and clustering.

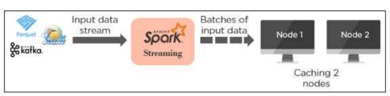


Ambari

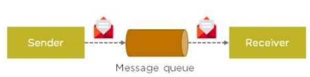
- Next up, we have Apache Ambari. It is an open-source tool responsible for keeping track of running applications and their statuses. Ambari manages, monitors, and provisions Hadoop clusters. Also, it also provides a central management service to start, stop, and configure Hadoop services.
- As seen in the following image, the Ambari web, which is your interface, is connected to the Ambari server. Apache Ambari follows a master/slave architecture. The master node is accountable for keeping track of the state of the infrastructure. For doing this, the master node uses a database server that can be configured during the setup time. Most of the time, the Ambari server is located on the MasterNode, and is connected to the database. This is where agents look into the host server. Agents run on all the nodes that you want to manage under Ambari. This program occasionally sends heartbeats to the master node to show its aliveness. By using Ambari Agent, the Ambari Server is able to execute many tasks.



Kafka

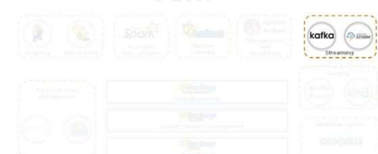


- Kafka uses a messaging system for transferring data from one application to another. As seen below, we have the sender, the message queue, and the receiver involved in data transfer.
- Kafka is a distributed streaming platform designed to store and process streams of records. It is written in Scala. It builds real-time streaming data pipelines that reliably get data between applications, and also builds real-time applications that transform data into streams.



Storm

- The storm is an engine that processes real-time streaming data at a very high speed. It is written in Clojure. A storm can handle over 1 million jobs on a node in a fraction of a second. It is integrated with Hadoop to harness higher throughputs.
- Now that we have looked at the various data ingestion tools and streaming services let us take a look at the security frameworks in the Hadoop ecosystem.



Ranger

- Ranger is a framework designed to enable, monitor, and manage data security across the Hadoop platform. It provides centralized administration for managing all security-related tasks. Ranger standardizes authorization across all Hadoop components, and provides enhanced support for different authorization methods like role-based access control, and attributes based access control, to name a few.

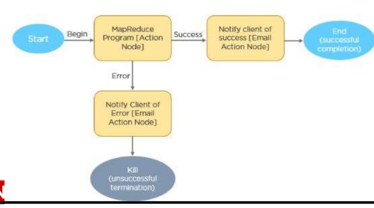
Knox

- Apache Knox is an application gateway used in conjunction with Hadoop deployments, interacting with REST APIs and UIs. The gateway delivers three types of user-facing services:
  - Proxying Services - This provides access to Hadoop via proxying the HTTP request
  - Authentication Services - This gives authentication for REST API access and WebSSO flow for user interfaces
  - Client Services - This provides client development either via scripting through DSL or using the Knox shell classes

Oozie

- Oozie is a workflow scheduler system used to manage Hadoop jobs. It consists of two parts:
  - Workflow engine - This consists of Directed Acyclic Graphs (DAGs), which specify a sequence of actions to be executed
  - Coordinator engine - The engine is made up of workflow jobs triggered by time and data availability

- As seen from the flowchart below, the process begins with the MapReduce jobs. This action can either be successful, or it can end in an error. If it is successful, the client is notified by an email. If the action is unsuccessful, the client is similarly notified, and the action is terminated.



THANK YOU
Q & A