CHAPTER 2

BIG DATA PLATFORMS - HADOOP

2.1 A BRIEF HISTORY OF HADOOP [31]

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 my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere: those are my naming criteria. Kids are good at generating such. Googol is a kid's term.

Sub projects 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 job racker keeps track of MapReduce jobs.

Building a web search engine from scratch was an ambitious goal, for not only is the software required to crawl and index websites complex to write, but it is also a challenge to run without a dedicated operations team, since there are so many moving parts. It's expensive, too: Mike Cafarella and Doug Cutting estimated a system supporting a 1-billion-page index would cost around half a million dollars in hardware, with a monthly running cost of $30,000.§ Nevertheless, they believed it was a worthy goal, as it would open up and ultimately democratize search engine algorithms.

Nutch was started in 2002, and a working crawler and search system quickly emerged. However, they realized that their architecture wouldn't scale to the billions of pages on the Web. Help was at hand with the publication of a paper in 2003 that described the architecture of Google's distributed filesystem, called GFS, which was being used in production at Google.GFS, or something like it, would solve their storage needs for the very large files generated as a part of the web crawl and indexing process. In particular, GFS would free up time being spent on administrative tasks such as managing storage nodes. In 2004, they set about writing an open source implementation, the Nutch Distributed Filesystem (NDFS).

In 2004, Google published the paper that introduced MapReduce to the world. Early in 2005, the Nutch developers had a working MapReduce implementation in Nutch, and by the middle of that year all the major Nutch algorithms had been ported to run using MapReduce and NDFS.

NDFS and the MapReduce implementation in Nutch were applicable beyond the realm of search, and in February 2006 they moved out of Nutch to form an independent subproject of Lucene called Hadoop. At around the same time, Doug Cutting joined Yahoo!, which provided a dedicated team and the resources to turn Hadoop into a system that ran at web scale (see sidebar). This was demonstrated in February 2008 when Yahoo! announced that its production search index was being generated by a 10,000-core Hadoop cluster.

In January 2008, Hadoop was made its own top-level project at Apache, confirming its success and its diverse, active community.By this time, Hadoop was being used by many other companies besides Yahoo!,such as Last.fm, Facebook, and the New York Times. Some applications are covered in the case studies in Chapter 16 and on the Hadoop wiki.

In one well-publicized feat, the New York Times used Amazon's EC2 compute cloud to crunch through four terabytes of scanned archives from the paper converting them to PDFs for the Web. The processing took less than 24 hours to run using 100 machines, and the project probably wouldn't have been embarked on without the combination of Amazon's pay-by-the-hour model (which allowed the NYT to access a large number of machines for a short period) and Hadoop's easy-to-use parallel programming model.

In April 2008, Hadoop broke a world record to become the fastest system to sort a terabyte of data. Running on a 910-node cluster, Hadoop sorted one terabyte in 209 seconds (just under 3½ minutes), beating the previous year's winner of 297 seconds (described in detail in "TeraByte Sort on Apache Hadoop"). In November of the same year, Google reported that its MapReduce implementation sorted one terabyte in 68 seconds. As the first edition of this book was going to press (May 2009), it was announced that a team at Yahoo! used Hadoop to sort one terabyte in 62 seconds.

2.2 BIG DATA & HADOOP – RESTAURANT ANALOGY [32]

Let us take an analogy of a restaurant to understand the problems associated with Big Data and how Hadoop solved that problem.

Bob is a businessman who has opened a small restaurant. Initially, in his restaurant, he used to receive two orders per hour and he had one chef with one food shelf in his restaurant which was sufficient enough to handle all the orders.

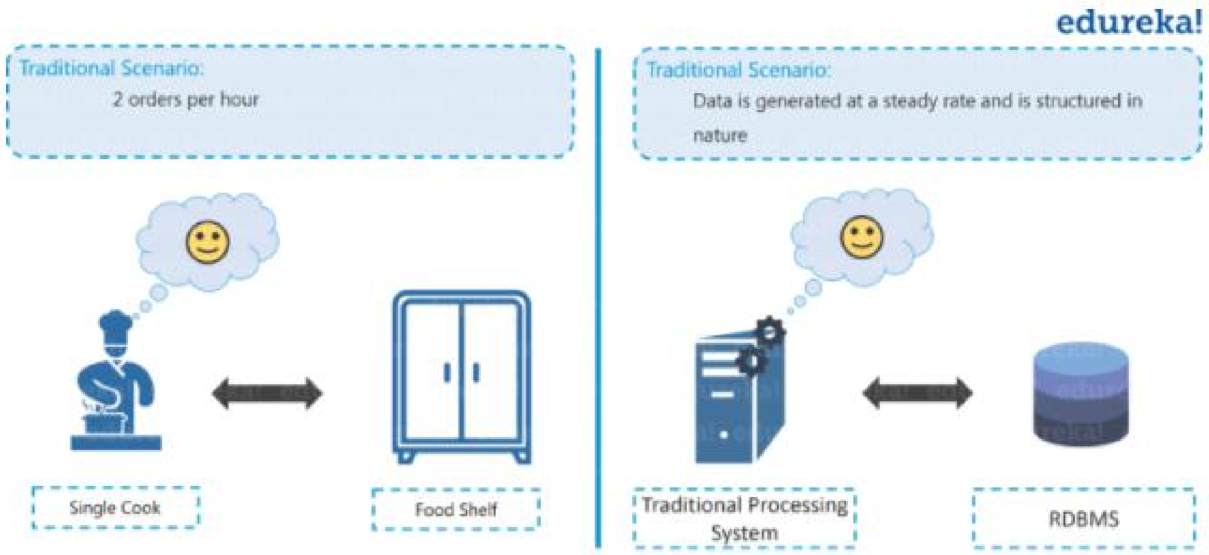


Figure 2.1 Traditional Restaurant Scenario [32]

Now let us compare the restaurant example with the traditional scenario where data was getting generated at a steady rate and our traditional systems like RDBMS is capable enough to handle it, just like Bob's chef. Here, you can relate the data storage with the restaurant's food shelf and the traditional processing unit with the chef as shown in the figure above.

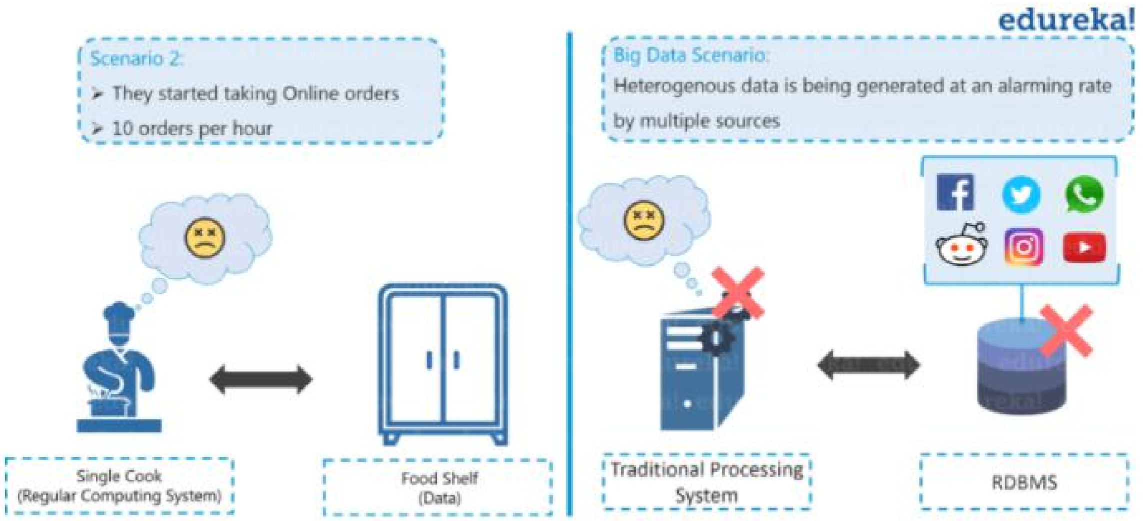

edureka!
Scenario 2:
They started taking Online orders
10 orders per hour
Big Data Scenario:
Heterogenous data is being generated at an alarming rate by multiple sources
Single Cook (Regular Computing System)
Food Shelf (Data)
Traditional Processing System
RDBMS

Figure 2.2 Traditional Scenario [32]

After few months, Bob thought of expanding his business and therefore, he started taking online orders and added few more cuisines to the restaurant's menu in order to engage a larger audience. Because of this transition, the rate at which they were receiving orders rose to an alarming figure of 10 orders per hour and it became quite difficult for a single cook to cope up with the current situation. Aware of the situation in processing the orders, Bob started thinking about the solution.

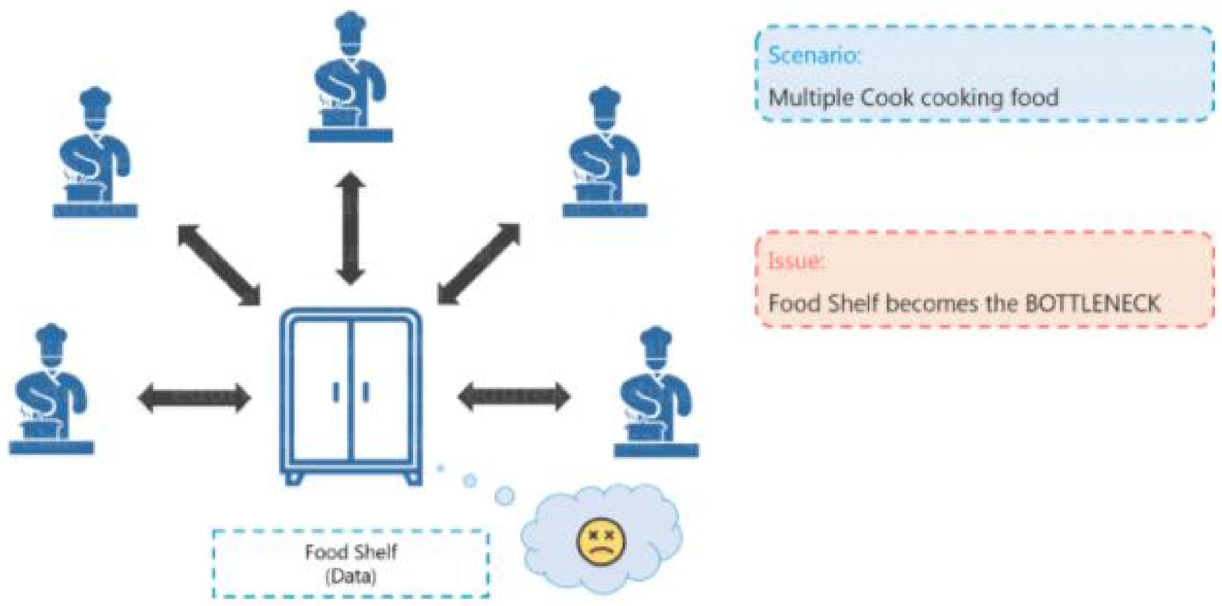


Figure 2.3 Distributed Processing Scenario [32]

Similarly, in Big Data scenario, the data started getting generated at an alarming rate because of the introduction of various data growth drivers such as social media, smartphones etc. Now, the traditional system, just like cook in Bob's restaurant, was not efficient enough to handle this sudden change. Thus, there was a need for a different kind of solutions strategy to cope up with this problem.

After a lot of research, Bob came up with a solution where he hired 4 more chefs to tackle the huge rate of orders being received. Everything was going quite well, but this solution led to one more problem. Since four chefs were sharing the same food shelf, the very food shelf was becoming the bottleneck of the whole process. Hence, the solution was not that efficient as Bob thought.

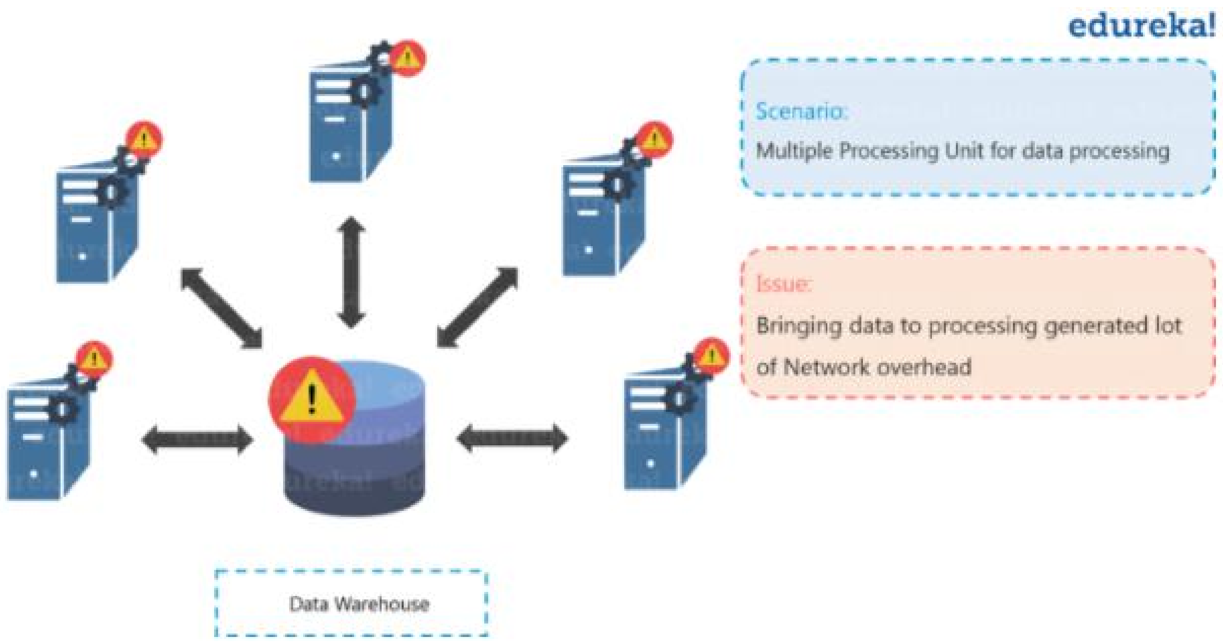

edureka!
Scenario:
Multiple Processing Unit for data processing
Issue:
Bringing data to processing generated lot of Network overhead
Data Warehouse

Figure 2.4 Distributed Processing Scenario Failure [32]

Similarly, to tackle the problem of processing huge datasets, multiple processing units were installed so as to process the data parallelly (just like Bob hired 4 chefs). But even in this case, bringing multiple processing units was not an effective solution because: the centralized storage unit became the bottleneck. In other words, the performance of the whole system is driven by the performance of the central storage unit. Therefore, the moment our central storage goes down, the whole system gets compromised. Hence, again there was a need to resolve this single point of failure.

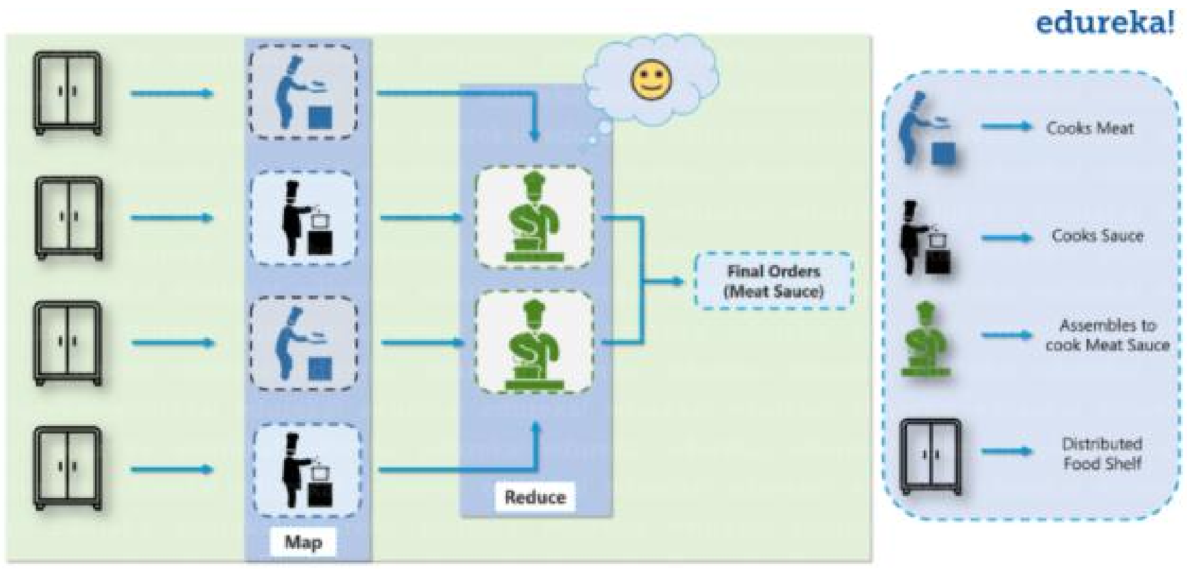


Figure 2.5 Solution to Restaurant Problem [32]

Bob came up with another efficient solution, he divided all the chefs in two hierarchies, i.e. junior and head chef and assigned each junior chef with a food shelf. Let us assume that the dish is Meat Sauce. Now, according to Bob's plan, one junior chef will prepare meat and the other junior chef will prepare the sauce. Moving ahead they will transfer both meat and sauce to the head chef, where the head chef will prepare the meat sauce after combining both the ingredients, which then will be delivered as the final order.

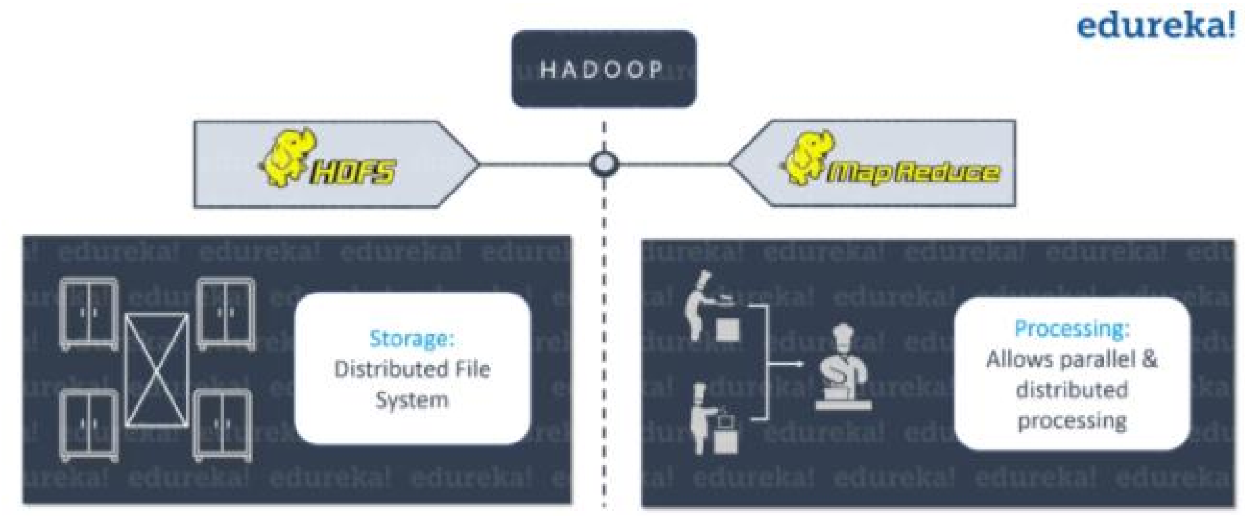


Figure 2.6 Hadoop in Restaurant Analogy [32]

Hadoop functions in a similar fashion as Bob's restaurant. As the food shelf is distributed in Bob's restaurant, similarly, in Hadoop, the data is stored in a distributed fashion with replications, to provide fault tolerance. For parallel processing, first the data is processed by the slaves where it is stored for some intermediate results and then those intermediate results are merged by master node to send the final result.

Now, you must have got an idea why Big Data is a problem statement and how Hadoop solves it. As we just discussed above, there were three major challenges with Big Data:

- ***The first problem is storing the colossal amount of data***. Storing huge data in a traditional system is not possible. The reason is obvious, the storage will be limited to one system and the data is increasing at a tremendous rate.
- ***The second problem is storing heterogeneous data***. Now we know that storing is a problem, but let me tell you it is just one part of the problem. The data is not only huge, but it is also present in various formats i.e. unstructured, semi-structured and structured. So, you need to make sure that you have a system to store different types of data that is generated from various sources.

- ***Finally let's focus on the third problem, which is the processing speed***. Now the time taken to process this huge amount of data is quite high as the data to be processed is too large.

To solve the storage issue and processing issue, two core components were created in Hadoop – HDFS and YARN. HDFS solves the storage issue as it stores the data in a distributed fashion and is easily scalable. And, YARN solves the processing issue by reducing the processing time drastically. Moving ahead, let us understand what is Hadoop?

2.3 WHAT IS HADOOP? [32]

Hadoop is an open-source software framework used for storing and processing Big Data in a distributed manner on large clusters of commodity hardware. Hadoop is licensed under the Apache v2 license. Hadoop was developed, based on the paper written by Google on MapReduce system and it applies concepts of functional programming. Hadoop is written in the Java programming language and ranks among the highest-level Apache projects. Hadoop was developed by Doug Cutting and Michael J. Cafarella.

2.3.1 Hadoop-as-a-Solution [32]

Let's understand how Hadoop provides solution to the Big Data problems that we have discussed so far.

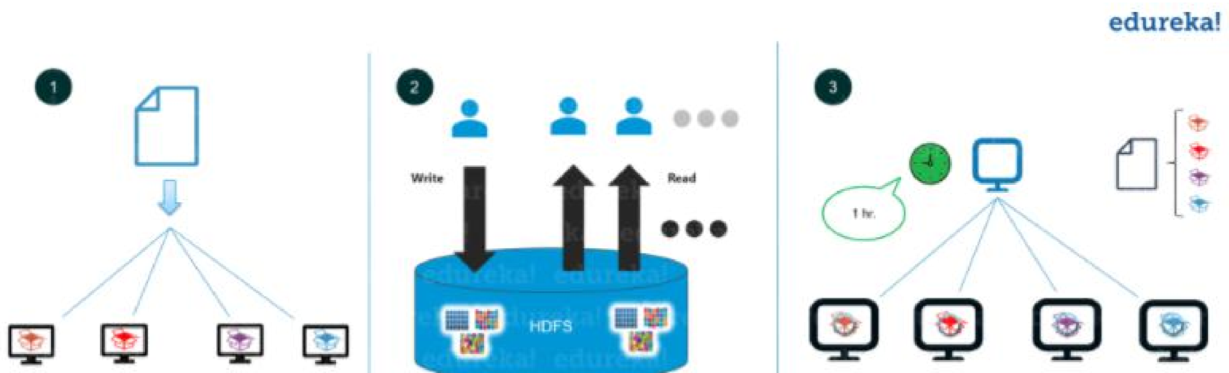


Figure 2.7 Hadoop-as-a-Solution [32]

The first problem is storing huge amount of data.

As you can see in the above image, HDFS provides a distributed way to store Big Data. Your data is stored in blocks in DataNodes and you specify the size of each block. Suppose you have 512MB of data and you have configured HDFS such that it will create 128 MB of data blocks. Now, HDFS will divide data into 4 blocks as 512/128=4 and stores it across different DataNodes. While storing these data blocks into DataNodes, data blocks are replicated on different DataNodes to provide fault tolerance.

Hadoop follows horizontal scaling instead of vertical scaling. In horizontal scaling, you can add new nodes to HDFS cluster on the run as per requirement, instead of increasing the hardware stack present in each node.

Next problem was storing the variety of data.
As you can see in the above image, in HDFS you can store all kinds of data whether it is structured, semi-structured or unstructured. In HDFS, there is no pre-dumping schema validation. It also follows write once and read many model. Due to this, you can just write any kind of data once and you can read it multiple times for finding insights.

The third challenge was about processing the data faster.
In order to solve this, we move processing unit to data instead of moving data to processing unit. So, what does it mean by moving the computation unit to data? It means that instead of moving data from different nodes to a single master node for processing, the processing logic is sent to the nodes where data is stored so as that each node can process a part of data in parallel. Finally, all of the intermediary output produced by each node is merged together and the final response is sent back to the client.

2.3.2 Hadoop Features [32]

Reliability:
When machines are working in tandem, if one of the machines fails, another machine will take over the responsibility and work in a reliable and fault tolerant fashion. Hadoop infrastructure has inbuilt fault tolerance features and hence, Hadoop is highly reliable.

Economical:
Hadoop uses commodity hardware (like your PC, laptop). For example, in a small Hadoop cluster, all your DataNodes can have normal configurations like 8-16 GB RAM with 5-10 TB hard disk and Xeon processors, but if I would have used hardware-based RAID with Oracle for the same purpose, I would end up spending 5x times more at least. So, the cost of ownership of a Hadoop-based project is pretty minimized. It is easier to maintain the Hadoop environment and is economical as well. Also, Hadoop is an open source software and hence there is no licensing cost.

Scalability:
Hadoop has the inbuilt capability of integrating seamlessly with cloud-based services. So, if you are installing Hadoop on a cloud, you don’t need to worry about the scalability

factor because you can go ahead and procure more hardware and expand your setup within minutes whenever required.

Flexibility:
Hadoop is very flexible in terms of ability to deal with all kinds of data. We discussed "Variety" in our previous blog on Big Data Tutorial, where data can be of any kind and Hadoop can store and process them all, whether it is structured, semi-structured or unstructured data.

These 4 characteristics make Hadoop a front-runner as a solution to Big Data challenges. Now that we know what is Hadoop, we can explore the core components of Hadoop. Let us understand, what are the core components of Hadoop.

2.3.3 Hadoop Core Components [32]

While setting up a Hadoop cluster, you have an option of choosing a lot of services as part of your Hadoop platform, but there are two services which are always mandatory for setting up Hadoop. One is HDFS (storage) and the other is YARN (processing). HDFS stands for Hadoop Distributed File System, which is a scalable storage unit of Hadoop whereas YARN is used to process the data i.e. stored in the HDFS in a distributed and parallel fashion.

HDFS
Let us go ahead with HDFS first. The main components of HDFS are: NameNode and DataNode. Let us talk about the roles of these two components in detail.

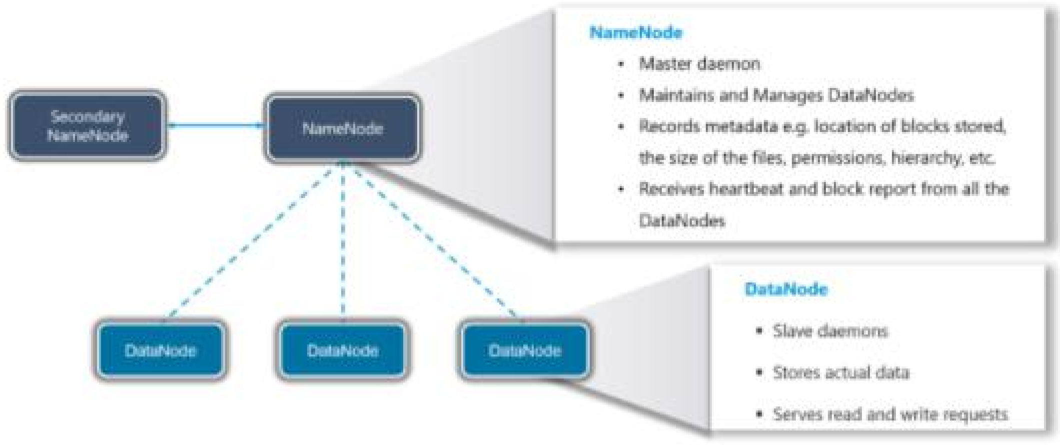

Secondary NameNode
NameNode
DataNode
DataNode
DataNode
NameNode
• Master daemon
• Maintains and Manages DataNodes
• Records metadata e.g. location of blocks stored, the size of the files, permissions, hierarchy, etc.
• Receives heartbeat and block report from all the DataNodes
DataNode
• Slave daemons
• Stores actual data
• Serves read and write requests

Figure 2.8 HDFS [32]

NameNode

- It is the master daemon that maintains and manages the DataNodes (slave nodes)
- It records the metadata of all the blocks stored in the cluster, e.g. location of blocks stored, size of the files, permissions, hierarchy, etc.
- It records each and every change that takes place to the file system metadata
- If a file is deleted in HDFS, the NameNode will immediately record this in the EditLog
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live
- It keeps a record of all the blocks in the HDFS and DataNode in which they are stored
- It has high availability and federation features which I will discuss in HDFS architecture in detail

DataNode

- It is the slave daemon which run on each slave machine
- The actual data is stored on DataNodes
- It is responsible for serving read and write requests from the clients
- It is also responsible for creating blocks, deleting blocks and replicating the same based on the decisions taken by the NameNode
- It sends heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds

So, this was all about HDFS in nutshell. Now, let move ahead to our second fundamental unit of Hadoop i.e. YARN.

YARN

YARN comprises of two major component: ResourceManager and NodeManager.

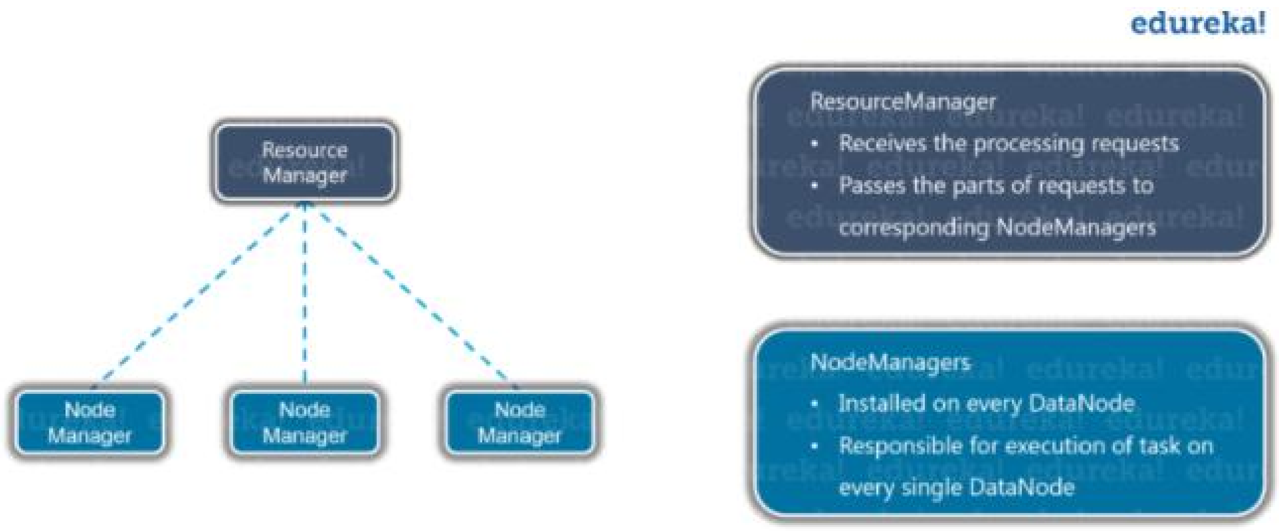

edureka!
Resource Manager
Node Manager
Node Manager
Node Manager
ResourceManager
• Receives the processing requests
• Passes the parts of requests to corresponding NodeManagers
NodeManagers
• Installed on every DataNode
• Responsible for execution of task on every single DataNode

Figure 2.9 YARN [32]

ResourceManager

- It is a cluster level (one for each cluster) component and runs on the master machine
- It manages resources and schedule applications running on top of YARN
- It has two components: Scheduler & ApplicationManager
- The Scheduler is responsible for allocating resources to the various running applications
- The ApplicationManager is responsible for accepting job submissions and negotiating the first container for executing the application
- It keeps a track of the heartbeats from the Node Manager

NodeManager

- It is a node level component (one on each node) and runs on each slave machine
- It is responsible for managing containers and monitoring resource utilization in each container
- It also keeps track of node health and log management
- It continuously communicates with ResourceManager to remain up-to-date

2.3.4 Hadoop Ecosystem [32]

So far you would have figured out that Hadoop is neither a programming language nor a service, it is a platform or framework which solves Big Data problems. You can consider it as a suite which encompasses a number of services for ingesting, storing and analyzing huge data sets along with tools for configuration management.

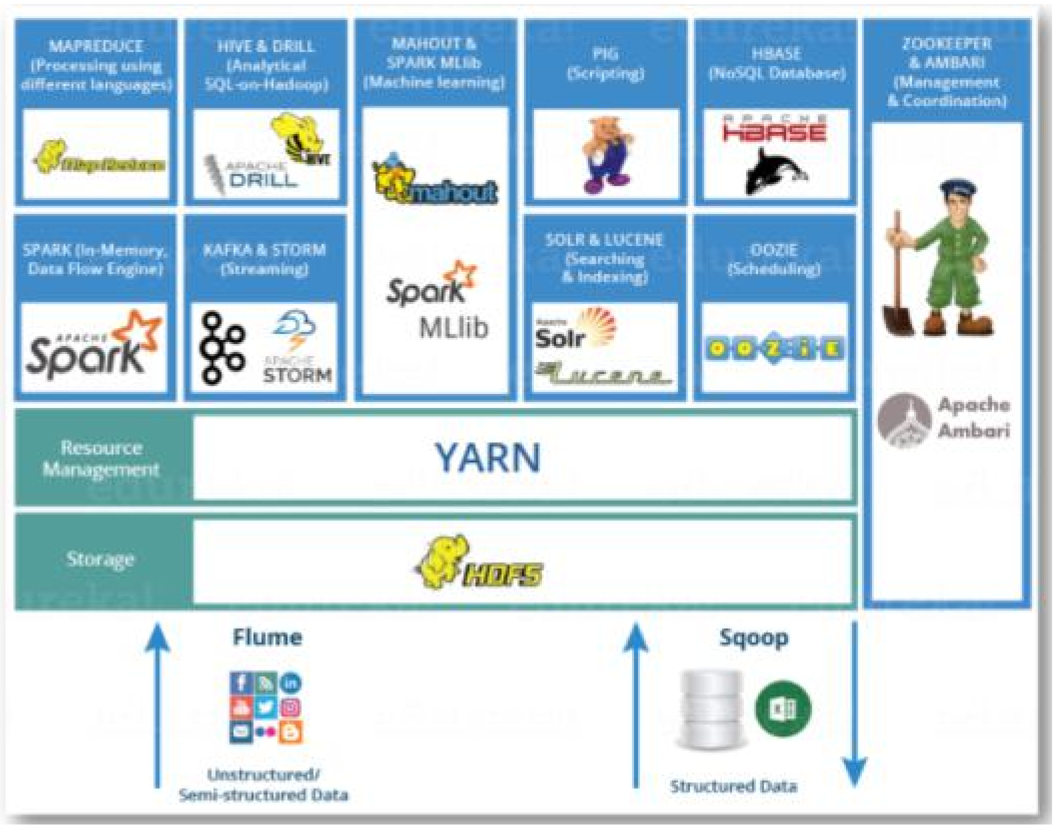

MAPREDUCE (Processing using different languages)
HIVE & DRILL (Analytical SQL-on-Hadoop)
MAHOUT & SPARK MLlib (Machine learning)
PIG (Scripting)
HBASE (NoSQL Database)
ZOOKEEPER & AMBARI (Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
KAFKA & STORM (Streaming)
SOLR & LUCENE (Searching & Indexing)
OOZIE (Scheduling)
Resource Management
YARN
Storage
HDFS
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data
Apache Ambari

Figure 2.9 Hadoop Ecosystem [32]

2.3.5 Examples of Hadoop - 5 Real-World Use Cases [33]

Here are five examples of Hadoop use cases:

1. Financial services companies use analytics to assess risk, build investment models, and create trading algorithms; Hadoop has been used to help build and run those applications.
2. Retailers use it to help analyze structured and unstructured data to better understand and serve their customers.
3. In the asset-intensive energy industry Hadoop-powered analytics are used for predictive maintenance, with input from Internet of Things (IoT) devices feeding data into big data programs.
4. Telecommunications companies can adapt all the aforementioned use cases. For example, they can use Hadoop-powered analytics to execute predictive maintenance on their infrastructure. Big data analytics can also plan efficient network paths and

recommend optimal locations for new cell towers or other network expansion. To support customer-facing operations telcos can analyze customer behavior and billing statements to inform new service offerings. Examples of Hadoop

5. There are numerous public sector programs, ranging from anticipating and preventing disease outbreaks to crunching numbers to catch tax cheats.

Hadoop is used in these and other big data programs because it is effective, scalable, and is well supported by large vendor and user communities. Hadoop is a de facto standard in big data.

2.3.6 Last.fm Case Study [32]

Last.fm is internet radio and community-driven music discovery service founded in 2002. Users transmit information to Last.fm servers indicating which songs they are listening to. The received data is processed and stored so that, the user can accesses it in the form of charts. Thus, Last.fm can make intelligent taste and compatible decisions for generating recommendations. The data is obtained from one of the two sources stated below:

- **scrobble**: When a user plays a track of his or her own choice and sends the information to Last.fm through a client application.
- **radio listen**: When the user tunes into a Last.fm radio station and streams a song.

Last.fm applications allow users to love, skip or ban each track they listen to. This track listening data is also transmitted to the server.

Last.fm applications allow users to love, skip or ban each track they listen to. This track listening data is also transmitted to the server.

- Over 40M unique visitors and 500M page views each month
- Scrobble stats:
    - Up to 800 scrobbles per second
    - More than 40 million scrobbles per day
    - Over 75 billion scrobbles so far
- Radio stats:
    - Over 10 million streaming hours per month
    - Over 400 thousand unique stations per day
- Each scrobble and radio listen generates at least one log line

Hadoop at Last.FM

- 100 Nodes

- 8 cores per node (dual quad-core)
- 24GB memory per node
- 8TB (4 disks of 2TB each)
- Hive integration to run optimized SQL queries for analysis

Last.FM started using Hadoop in 2006 because of the growth in users from thousands to millions. With the help of Hadoop they processed hundreds of daily, monthly, and weekly jobs including website stats and metrics, chart generation (i.e. track statistics), metadata corrections (e.g. misspellings of artists), indexing for search, combining/formatting data for recommendations, data insights, evaluations & reporting. This helped Last.FM to grow tremendously and figure out the taste of their users, based on which they started recommending music.

2.3.7 Advantages of using Hadoop [34]

Hadoop helps organizations make decisions based on comprehensive analysis of multiple variables and data sets, rather than a small sampling of data or anecdotal incidents. The ability to process large sets of disparate data gives Hadoop users a more comprehensive view of their customers, operations, opportunities, risks, etc. To develop a similar perspective without big data, organizations would need to conduct multiple, limited data analyses then find a way to synthesize the results, which would likely involve a lot of manual effort and subjective analysis.

Here are some other benefits to using Hadoop:

- **Advanced data analysis can be done in-house** – Hadoop makes it practical to work with large data sets and customize the outcome without having to outsource the task to specialist service providers. Keeping operations in-house helps organizations be more agile, while also avoiding the ongoing operational expense of outsourcing. Hadoop Advantages
- **Organizations can fully leverage their data** – One alternative to not using Hadoop is simply not to use all the data and inputs that are available to support business activity. With Hadoop organizations can take full advantage of all their data – structured and unstructured, real-time and historical. Leveraging adds more value to the data itself and improves the return on investment (ROI) for the legacy systems used to collect, process, and store the data, including ERP and CRM systems, social media programs, sensors, industrial automation systems, etc.
- **Run a commodity vs. custom architecture** – Some of the tasks that Hadoop is being used for today were formerly run by MPCC and other specialty, expensive

computer systems. Hadoop commonly runs on commodity hardware. Because it is the de facto big data standard, it is supported by a large and competitive solution provider community, which protects customers from vendor lock-in.

How Hadoop's fundamental problem solving capabilities are applied depends on the use case. As noted, Hadoop data processing is commonly used to support better decision making, provide real-time monitoring for things like machine conditions, threat levels, transaction volumes, etc., and to enable predictive, proactive activity.

It's a thing – Hadoop and the Internet of Things (IoT)

Hadoop can be an important enabling technology for Internet of Things (IoT) projects. In a typical IoT application, a network of sensors or other intelligent devices continually sends current condition data to a platform solution that parses the data, processes what is relevant, and automatically directs an appropriate action (such as shutting down a machine that is at risk of overheating). The sensor and device data is also stored for additional analytics.

IoT programs often produce volumes and types of data that enterprises have never dealt with before. For example, an intelligent factory system can produce millions of sensor readings each day. Organizations that pursue IoT may be quickly pulled into the world of big data. Hadoop can provide a lifeline for efficiently storing, processing, and managing the new data sources.

2.3.8 Disadvantages of using Hadoop [34]

Despite its popularity Hadoop is still an emerging technology, and many of its limitations relate to its newness. The by-products of Hadoop's rapid expansion and evolution include skills gaps, a lack of complementary solutions to support specific needs (e.g., development and debugging tools, native Hadoop support in specific software solutions, etc.). Other criticism stems from Hadoop's status as an open source project, as some professionals consider open source too unstable for business. Other critics say Hadoop is better for storing and aggregating data than it is for processing it.

These disadvantages could be eased as Hadoop becomes more mature. Notably, some of the aforementioned criticisms are really a reflection of the limitations of open-source solutions embedded in the Hadoop ecosystem. For example, the Oozie workflow scheduling utility is often cited as limited and inconvenient to work with, but there are third-party solutions that overcome its limitations and, in some cases, eliminate the need to use Oozie at all.

In addition, Hadoop has some fundamental characteristics that limit its capabilities. Here are some of the most-cited limitations and criticisms regarding Hadoop.

- **Storage requirements** – Hadoop's built-in redundancy duplicates data, thereby requiring more storage resources.
- **Limited SQL support** – Hadoop lacks some of the query functions that SQL database users are accustomed to.
- **Limited native security** – Hadoop does not encrypt data while in storage or when on the network. Further, Hadoop is based on Java, which is a frequent target for malware and other hacks.
- **Component limitations** – There are multiple specific criticisms regarding limitations of Hadoop's four core components (HDFS, YARN, MapReduce and Common). Some of these limitations are overcome by third-party solutions, but the functionality is lacking in Hadoop itself.

2.3.9 Is Hadoop an efficient use of resources? [34]

Once organizations determine that Hadoop will give them a way to work with big data, they often begin to wonder if it is an efficient way. In most cases the answer is yes. Hadoop is also often more cost effective and resource efficient than the methods that are commonly used to maintain enterprise data warehouses (EDWs).

Hadoop is an efficient and cost effective platform for big data because it runs on commodity servers with attached storage, which is a less expensive architecture than a dedicated storage area network (SAN). Commodity Hadoop clusters are also very scalable, which is important because big data programs tend to get bigger as users gain experience and business value from them.

Hadoop not only makes it cost effective to work the big data, but also reduces the costs of maintaining an existing enterprise data warehouse. That's because the essential extract-transport-load (ETL) tasks that are typically performed on the EDW hardware can be offloaded for execution on lower-cost Hadoop clusters. ETL takes a lot of processing cycles, so it is more resource efficient not to execute them on the high-end machines where enterprise data warehouses reside.

The cost and value for using Hadoop depends on the use cases, specific tools and configurations used, and the amount of data in the environment. The supporting tools and solutions used along with core Hadoop technology have a tremendous influence on the costs to develop and maintain the environment. Deeper insight into the Hadoop ecosystem is provided in the following section.

2.3.10 The business case for Hadoop [34]

The business case for Hadoop depends on the value of information. Hadoop can make more information available, it can analyze more information to support better decision making, and it can make information more valuable by analyzing it in real time. Unlike earlier-generation business intelligence technology, Hadoop helps organizations make sense of both structured and unstructured data. Hadoop can produce new insights because it is able to process new data types, such as social media streams and sensor input. A wide range of organizations have found value in Hadoop by using it to help them better understand their customers, competitors, supply chains, risks and opportunities.

Hadoop can help you take advantage of information from the past, present, and future. Many people think big data is mostly focused on analyzing historical information, which may or may not be relevant in current conditions. While big data solutions often do historical analysis to make predictions and recommendations, it also provides a means to analyze and act on current condition data in real time. For example, Hadoop is the foundation for the recommendation engines some e-commerce retailers use to suggest items while customers are browsing their websites. The recommendations are based on analysis of the specific customer's previous purchase history, what other customers purchased along with the item being viewed, and what's currently trending by performing sentiment analysis on input from social media streams. Those multiple data sources must be processed, analyzed, and converted into actionable information in the short time the customer is on the page. Hadoop makes it all possible, and retailers are reporting sales lifts of between 2 percent and 20 percent as a result. When the sale is made, Hadoop helps financial institutions detect and prevent potential credit card fraud in real time.

Hadoop-driven solutions can also consider current and historical data to guide future activity, such as making assortment planning recommendations for retailers or developing predictive maintenance schedules for production equipment and other assets.

2.3.11 Another way to look at value [34]

The business case for Hadoop is different for every business. To begin to understand whether Hadoop is a fit for your organization and how it could provide value, ask:

- What is the value of better decision making?

- Do past customer behaviors, business conditions, risk factors, etc. have any bearing on current or future performance?
- Would faster decision-making help the business?
- Would predictive maintenance and reducing unplanned downtime be valuable?
- What incremental value would improve demand forecasting provide?
- Could we benefit from sentiment analysis?
- Would real-time monitoring assist information security or compliance requirements?
- Are we getting all we can out of the structured and unstructured data we have?

Hadoop's use cases and benefits are very broad. In many cases, Hadoop won't be introduced as a replacement technology, but instead will be used to do things that haven't been done before. Therefore it is helpful to examine Hadoop's benefits, limitations, and alternatives from a business, not technical, perspective.

2.3.12 What does Hadoop replace? [34]

Hadoop is commonly used to support better decision making. Therefore it often complements the data processing and reporting systems that are already used, rather than replacing them. For example, Tableau is a popular business intelligence tool that organizations use to process and visualize a variety of data. Tableau supports Hadoop and provides another option to output Hadoop-driven data analysis. Excel can also process data imported from Hadoop, however, more sophisticated, big data-oriented solutions support more capabilities. The Hadoop ecosystem is continually expanding with new solutions to help users take advantage of Hadoop in new ways.

Hadoop does replace the need to have clusters of customized, high-performance computers that are supported by large teams of technicians and data scientists to turn data into actionable information. Hadoop is an alternative to using massively parallel processing (MPP) database structures, which tend to be custom built and expensive. Hadoop can also replace traditional silo-based IT architectures, or at least provide a way for the silos to interact and serve as a single source for data.

2.3.13 Problems that Hadoop solves [34]

Hadoop solves the fundamental problem of processing large sets of structured and unstructured data that come from disparate sources and reside in different systems. More specifically, Hadoop addresses the challenges of scale, latency, and comprehensiveness when dealing with large data sets.

Scale – Hadoop makes it practical to process large sets of data on commodity computers. It also solves the problem of getting an accurate, single view of structured and unstructured data that is held in multiple systems.
Latency – Hadoop can process large data sets much faster than traditional methods, so organizations can act on the data while it is still current. For example, recommendation engines suggest complementary items or special offers in real time. In the past, organizations might have run a report on what customers purchased, analyzed the results, then sent a follow-up email days or weeks later suggesting the same complementary items. There is also tremendous value to removing latency when monitoring for network intrusion, financial fraud and other threats.
Comprehensiveness - One of the things that set Hadoop apart is its ability to process different types of data. Besides traditional, structured data, sometimes referred to as data at rest, Hadoop can sort and process data in motion, such as input from sensors, location data, social media channels, and metadata from video and customer contact systems. It can also perform clickstream data analysis. The comprehensiveness creates more visibility and a deeper understanding of what is being studied.

These Hadoop characteristics have also been expressed as the three Vs – volume, velocity, and variety.

2.4 HADOOP INSTALLATION [35]

Environment required for Hadoop: The production environment of Hadoop is UNIX, but it can also be used in Windows using Cygwin. Java 1.6 or above is needed to run Map Reduce Programs. For Hadoop installation from tar ball on the UNIX environment you need

1. Java Installation
2. SSH installation
3. Hadoop Installation and File Configuration

2.4.1 Java Installation [35]

Step 1. Type "java -version" in prompt to find if the java is installed or not. If not then download java from http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html . The tar filejdk-7u71-linux-x64.tar.gz will be downloaded to your system.

Step 2. Extract the file using the below command

```
#tar zxf jdk-7u71-linux-x64.tar.gz
```

Step 3. To make java available for all the users of UNIX move the file to /usr/local and set the path. In the prompt switch to root user and then type the command below to move the jdk to /usr/lib.

```
# mv jdk1.7.0_71 /usr/lib/
```

Now in ~/.bashrc file add the following commands to set up the path.

```
# export JAVA_HOME=/usr/lib/jdk1.7.0_71
# export PATH=PATH:$JAVA_HOME/bin
```

Now, you can check the installation by typing "java -version" in the prompt.

2.4.2 SSH Installation [35]

SSH is used to interact with the master and slaves computer without any prompt for password. First of all create a Hadoop user on the master and slave systems

```
# useradd hadoop
# passwd Hadoop
```

To map the nodes open the hosts file present in /etc/ folder on all the machines and put the ip address along with their host name.

```
# vi /etc/hosts
```

Enter the lines below

```
190.12.1.114    hadoop-master
190.12.1.121    hadoop-salve-one
190.12.1.143   hadoop-slave-two
```

Set up SSH key in every node so that they can communicate among themselves without password. Commands for the same are:

```
# su hadoop
$ ssh-keygen -t rsa
$ ssh-copy-id -i ~/.ssh/id_rsa.pub tutorialspoint@hadoop-master
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop_tp1@hadoop-slave-1
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop_tp2@hadoop-slave-2
$ chmod 0600 ~/.ssh/authorized_keys
$ exit
```

2.4.3 Hadoop Installation [35]

Hadoop can be downloaded from http://developer.yahoo.com/hadoop/tutorial/module3.html

Now extract the Hadoop and copy it to a location.

```
$ mkdir /usr/hadoop
$ sudo tar vxzf  hadoop-2.2.0.tar.gz ?c /usr/hadoop
```

Change the ownership of Hadoop folder

```
$sudo chown -R hadoop  usr/hadoop
```

Change the Hadoop configuration files:

All the files are present in /usr/local/Hadoop/etc/hadoop

1) In hadoop-env.sh file add

```
export JAVA_HOME=/usr/lib/jvm/jdk/jdk1.7.0_71
```

2) In core-site.xml add following between configuration tabs,

```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://hadoop-master:9000</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
</configuration>
```

3) In hdfs-site.xmladd following between configuration tabs,

```
<configuration>
<property>
<name>dfs.data.dir</name>
<value>usr/hadoop/dfs/name/data</value>
<final>true</final>
</property>
<property>
<name>dfs.name.dir</name>
<value>usr/hadoop/dfs/name</value>
<final>true</final>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
```

4) Open the Mapred-site.xml and make the change as shown below

```
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>hadoop-master:9001</value>
</property>
</configuration>
```

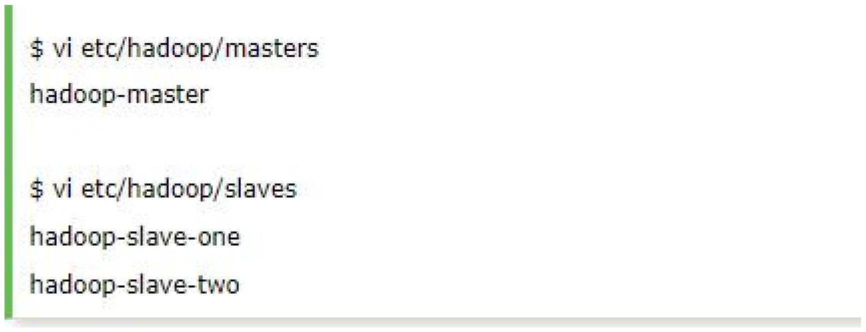
5) Finally, update your $HOME/.bahsrc

```
cd $HOME
vi .bashrc
Append following lines in the end and save and exit
#Hadoop variables
export JAVA_HOME=/usr/lib/jvm/jdk/jdk1.7.0_71
export HADOOP_INSTALL=/usr/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
```

On the slave machine install Hadoop using the command below

```
# su hadoop
$ cd /opt/hadoop
$ scp -r hadoop hadoop-slave-one:/usr/hadoop
$ scp -r hadoop hadoop-slave-two:/usr/Hadoop
```

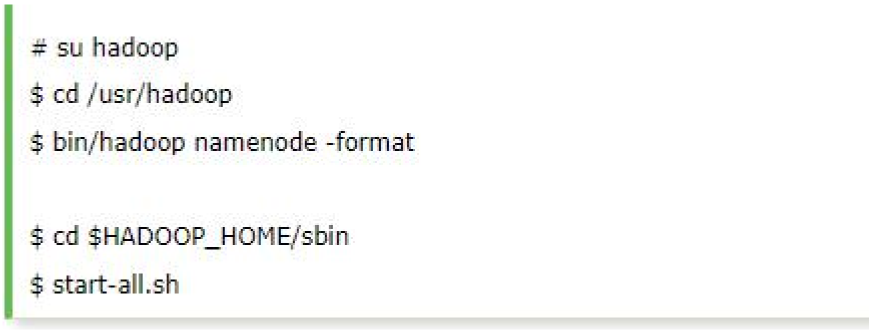
Configure master node and slave node



```
$ vi etc/hadoop/masters
hadoop-master

$ vi etc/hadoop/slaves
hadoop-slave-one
hadoop-slave-two
```

After this format the name node and start all the deamons



```
# su hadoop
$ cd /usr/hadoop
$ bin/hadoop namenode -format

$ cd $HADOOP_HOME/sbin
$ start-all.sh
```

The easiest step is the usage of cloudera as it comes with all the stuffs pre-installed which can be downloaded from http://content.udacity-data.com/courses/ud617/Cloudera-Udacity-Training-VM-4.1.1.c.zip

2.5 HADOOP MODULES [35]

2.5.1 HDFS [35]

What is HDFS

Hadoop comes with a distributed file system called HDFS. In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application.

It is cost effective as it uses commodity hardware. It involves the concept of blocks, data nodes and node name.

Where to use HDFS

- **Very Large Files**: Files should be of hundreds of megabytes, gigabytes or more.

- **Streaming Data Access**: The time to read whole data set is more important than latency in reading the first. HDFS is built on write-once and read-many-times pattern.
- **Commodity Hardware**: It works on low cost hardware.

Where not to use HDFS

- **Low Latency data access**: Applications that require very less time to access the first data should not use HDFS as it is giving importance to whole data rather than time to fetch the first record.
- **Lots Of Small Files**: The name node contains the metadata of files in memory and if the files are small in size it takes a lot of memory for name node's memory which is not feasible.
- **Multiple Writes**: It should not be used when we have to write multiple times.

HDFS Concepts

1. **Blocks**: A Block is the minimum amount of data that it can read or write.HDFS blocks are 128 MB by default and this is configurable.Files n HDFS are broken into block-sized chunks,which are stored as independent units.Unlike a file system, if the file is in HDFS is smaller than block size, then it does not occupy full block?s size, i.e. 5 MB of file stored in HDFS of block size 128 MB takes 5MB of space only.The HDFS block size is large just to minimize the cost of seek.

2. **Name Node**: HDFS works in master-worker pattern where the name node acts as master.Name Node is controller and manager of HDFS as it knows the status and the metadata of all the files in HDFS; the metadata information being file permission, names and location of each block.The metadata are small, so it is stored in the memory of name node,allowing faster access to data. Moreover the HDFS cluster is accessed by multiple clients concurrently,so all this information is handled bya single machine. The file system operations like opening, closing, renaming etc. are executed by it.

3. **Data Node**: They store and retrieve blocks when they are told to; by client or name node. They report back to name node periodically, with list of blocks that they are storing. The data node being commodity hardware also does the work of block creation, deletion and replication as stated by the name node.

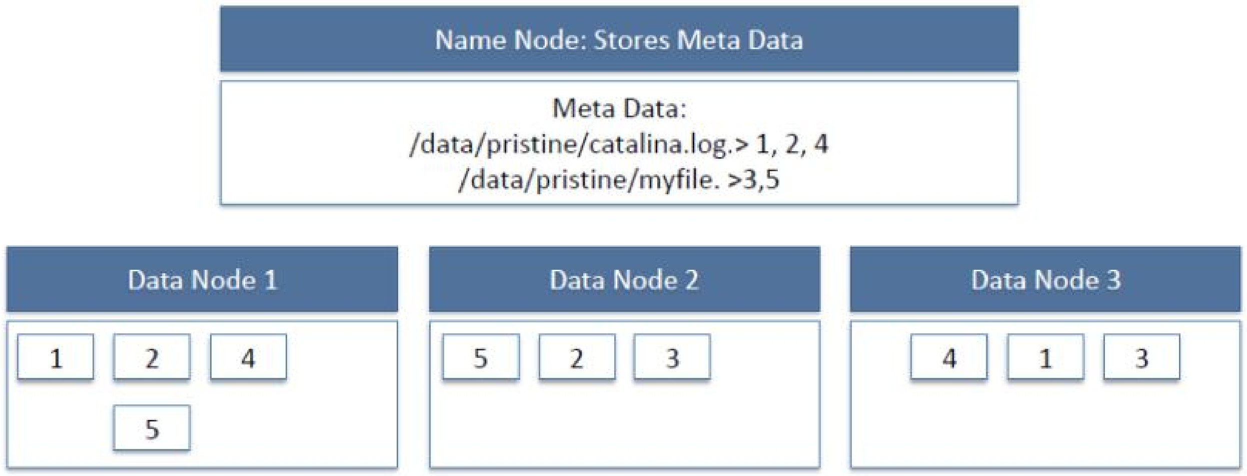

Name Node: Stores Meta Data
Meta Data:
/data/pristine/catalina.log.> 1, 2, 4
/data/pristine/myfile. >3,5
Data Node 1
1
2
4
5
Data Node 2
5
2
3
Data Node 3
4
1
3

Figure 2.10 HDFS DataNode and NameNode Image [35]

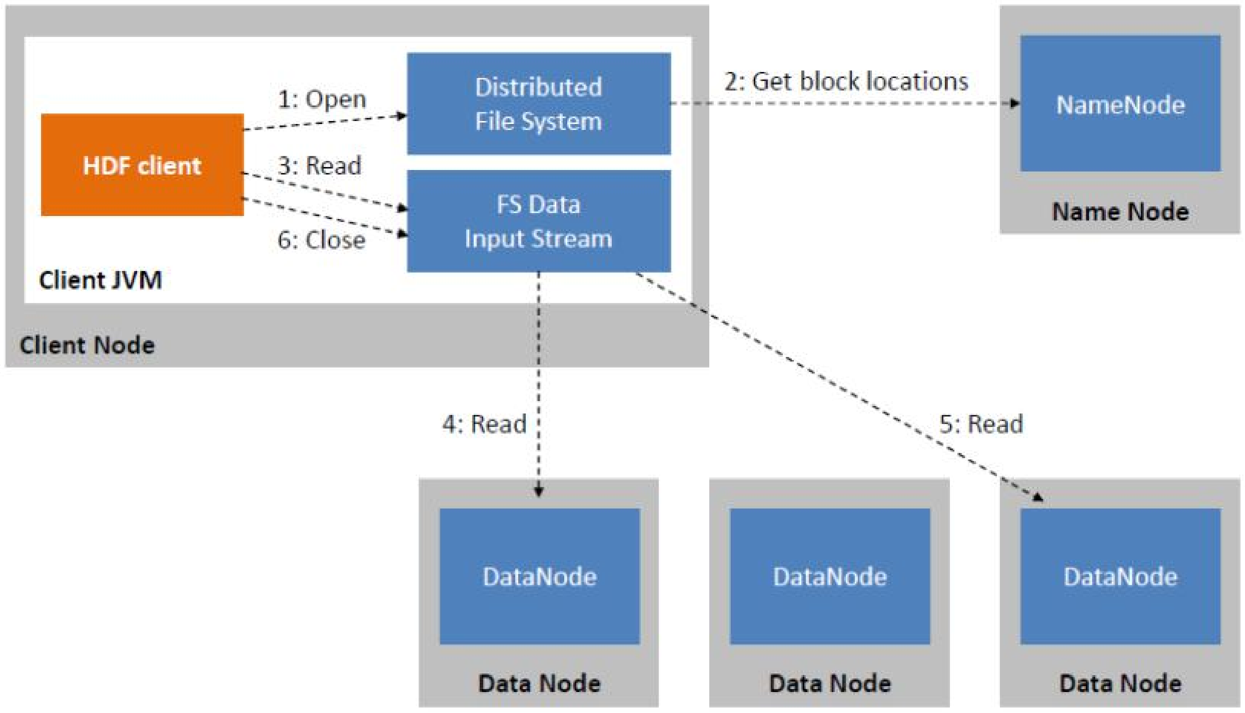


Figure 2.11 HDFS Read Image [35]

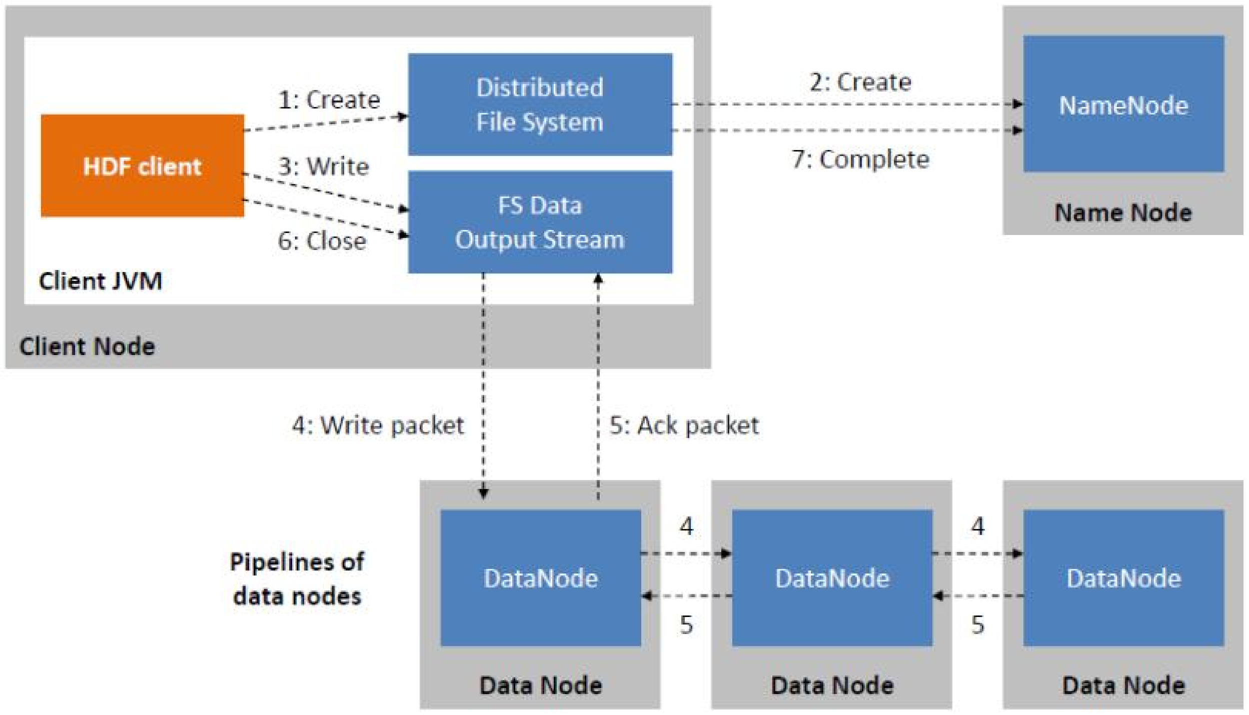


Figure 2.12 HDFS Write Image [35]

Since all the metadata is stored in name node, it is very important. If it fails the file system cannot be used as there would be no way of knowing how to reconstruct the files from blocks present in data node. To overcome this, the concept of secondary name node arises.

Secondary Name Node: It is a separate physical machine which acts as a helper of name node. It performs periodic check points. It communicates with the name node and take snapshot of meta data which helps minimize downtime and loss of data.

Starting HDFS

The HDFS should be formatted initially and then started in the distributed mode. Commands are given below.

To Format **$ hadoop namenode -format**

To Start **$ start-dfs.sh**

HDFS Basic File Operations

1. Putting data to HDFS from local file system
    - First create a folder in HDFS where data can be put form local file system.
      $ hadoop fs -mkdir /user/test

    - Copy the file "data.txt" from a file kept in local folder /usr/home/Desktop to HDFS folder /user/ test
      $ hadoop fs -copyFromLocal /usr/home/Desktop/data.txt /user/test

    - Display the content of HDFS folder
      $ Hadoop fs -ls /user/test

2. Copying data from HDFS to local file system
    - $ hadoop fs -copyToLocal /user/test/data.txt /usr/bin/data_copy.txt

3. Compare the files and see that both are same
    - $ md5 /usr/bin/data_copy.txt /usr/home/Desktop/data.txt

Recursive deleting

- hadoop fs -rmr <arg>

Example:

- hadoop fs -rmr /user/sonoo/

HDFS Other commands

The below is used in the commands

"<path>" means any file or directory name.

"<path>..." means one or more file or directory names.

"<file>" means any filename.

"<src>" and "<dest>" are path names in a directed operation.

"<localSrc>" and "<localDest>" are paths as above, but on the local file system

- put <localSrc><dest>
  Copies the file or directory from the local file system identified by localSrc to dest within the DFS.

- copyFromLocal <localSrc><dest>
  Identical to -put

- copyFromLocal <localSrc><dest>
  Identical to -put

- moveFromLocal <localSrc><dest>
  Copies the file or directory from the local file system identified by localSrc to dest within HDFS, and then deletes the local copy on success.

- get [-crc] <src><localDest>
  Copies the file or directory in HDFS identified by src to the local file system path identified by localDest.

- cat <filen-ame>
  Displays the contents of filename on stdout.

- moveToLocal <src><localDest>
  Works like -get, but deletes the HDFS copy on success.

- setrep [-R] [-w] rep <path>
  Sets the target replication factor for files identified by path to rep. (The actual replication factor will move toward the target over time)

- touchz <path>
  Creates a file at path containing the current time as a timestamp. Fails if a file already exists at path, unless the file is already size 0.

- test -[ezd] <path>
  Returns 1 if path exists; has zero length; or is a directory or 0 otherwise.

- stat [format] <path>
  Prints information about path. Format is a string which accepts file size in blocks (%b), filename (%n), block size (%o), replication (%r), and modification date (%y, %Y).

2.5.2 YARN [35]

What is YARN

Yet Another Resource Manager takes programming to the next level beyond Java , and makes it interactive to let another application Hbase, Spark etc. to work on it.Different Yarn applications can co-exist on the same cluster so MapReduce, Hbase, Spark all can run at the same time bringing great benefits for manageability and cluster utilization.

Components of YARN

- **Client**: For submitting MapReduce jobs.
- **Resource Manager**: To manage the use of resources across the cluster
- **Node Manager**: For launching and monitoring the computer containers on machines in the cluster.
- **Map Reduce Application Master**: Checks tasks running the MapReduce job. The application master and the MapReduce tasks run in containers that are scheduled by the resource manager, and managed by the node managers.

Jobtracker & Tasktrackerwere were used in previous version of Hadoop, which were responsible for handling resources and checking progress management. However, Hadoop 2.0 has Resource manager and NodeManager to overcome the shortfall of Jobtracker & Tasktracker.

Benefits of YARN

- **Scalability**: Map Reduce 1 hits ascalability bottleneck at 4000 nodes and 40000 task, but Yarn is designed for 10,000 nodes and 1 lakh tasks.
- **Utiliazation**: Node Manager manages a pool of resources, rather than a fixed number of the designated slots thus increasing the utilization.
- **Multitenancy**: Different version of MapReduce can run on YARN, which makes the process of upgrading MapReduce more manageable.

2.5.3 MapReduce [35]

To take the advantage of parallel processing of Hadoop, the query must be in MapReduce form. The MapReduce is a paradigm which has two phases, the mapper phase and the reducer phase. In the Mapper the input is given in the form of key value pair. The output of the mapper is fed to the reducer as input. The reducer runs only after the mapper is over. The reducer too takes input in key value format and the output of reducer is final output.

Steps in Map Reduce

- Map takes a data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.
- Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.
- Output of sort and shuffle will be sent to reducer phase. Reducer will perform a defined function on list of values for unique keys and Final output will<key, value> will be stored/displayed.

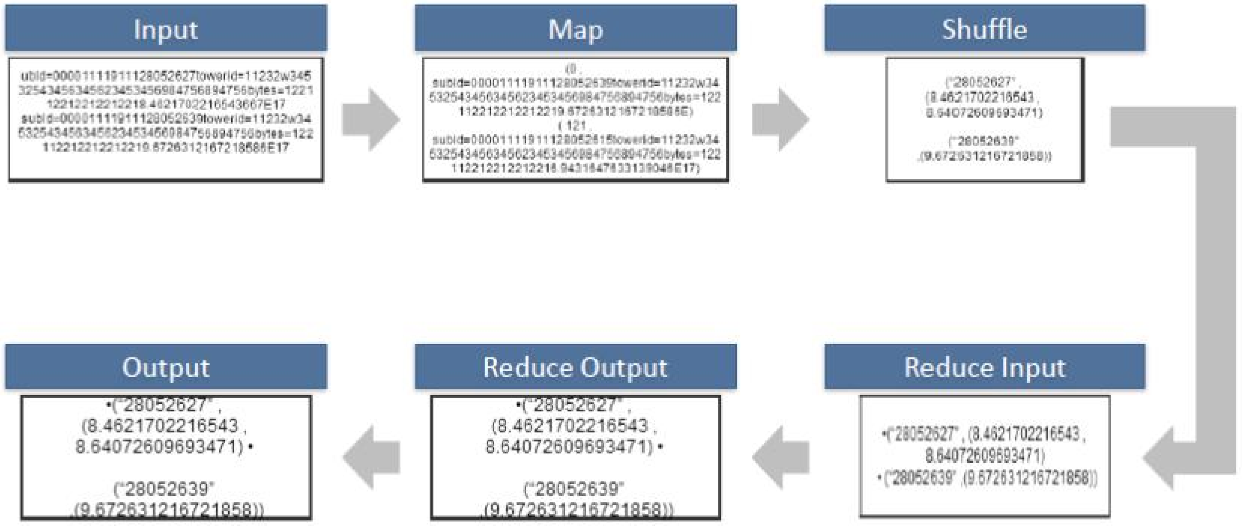

Input
ubId=00001111911128052627towerId=11232w34532543456345623453456984756894756bytes=122112212212218.4621702216543667E17 subId=00001111911128052639towerId=11232w34632543456345623453456984756894756bytes=122112212212219.6726312167218586E17
Map
(0, subId=00001111911128052639towerId=11232w34632543456345623453456984756894756bytes=122112212212219.6726312167218586E)
(121, subId=00001111911128052615towerId=11232w34632543456345623453456984756894756bytes=122112212212218.9431647833139048E17)
Shuffle
("28052627", (8.4621702216543, 8.64072609693471)
("28052639", (9.672631216721858))
Reduce Input
•("28052627", (8.4621702216543, 8.64072609693471)
•("28052639", (9.672631216721858))
Reduce Output
•("28052627", (8.4621702216543, 8.64072609693471) •
("28052639", (9.672631216721858))
Output
•("28052627", (8.4621702216543, 8.64072609693471) •
("28052639", (9.672631216721858))

Figure 2.13 Steps in Map Reduce [35]

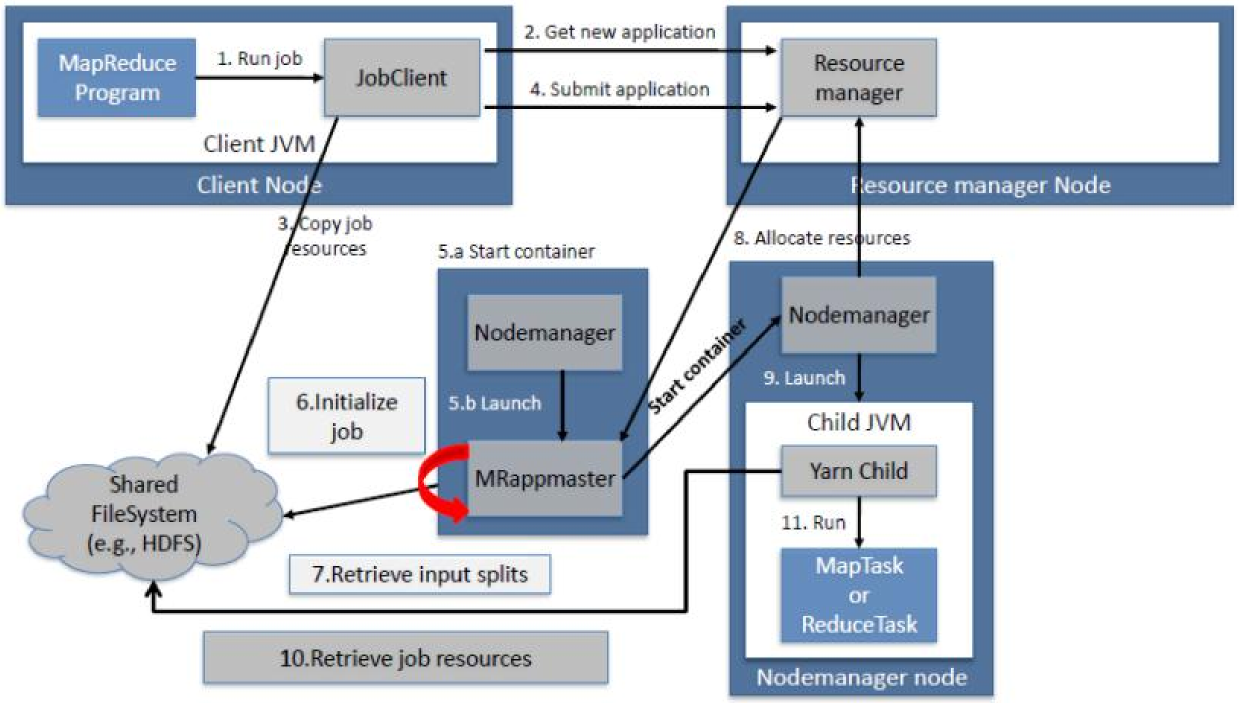


Figure 2.14 Map Reduce Architecture [35]

How Many Maps

The size of data to be processed decides the number of maps required. For example, we have 1000 MB data and block size is 64 MB then we need 16 mappers.

Sort and Shuffle

The sort and shuffle occur on the output of mapper and before the reducer. When the mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk. Using the input from each mapper <k2,v2> , we collect all the values for each unique key k2. This output from the shuffle phase in the form of <k2,list(v2)> is sent as input to reducer phase.

MapReduce Example

Use Case

Find the number of occurrences of the word using Map Reduce in a text file

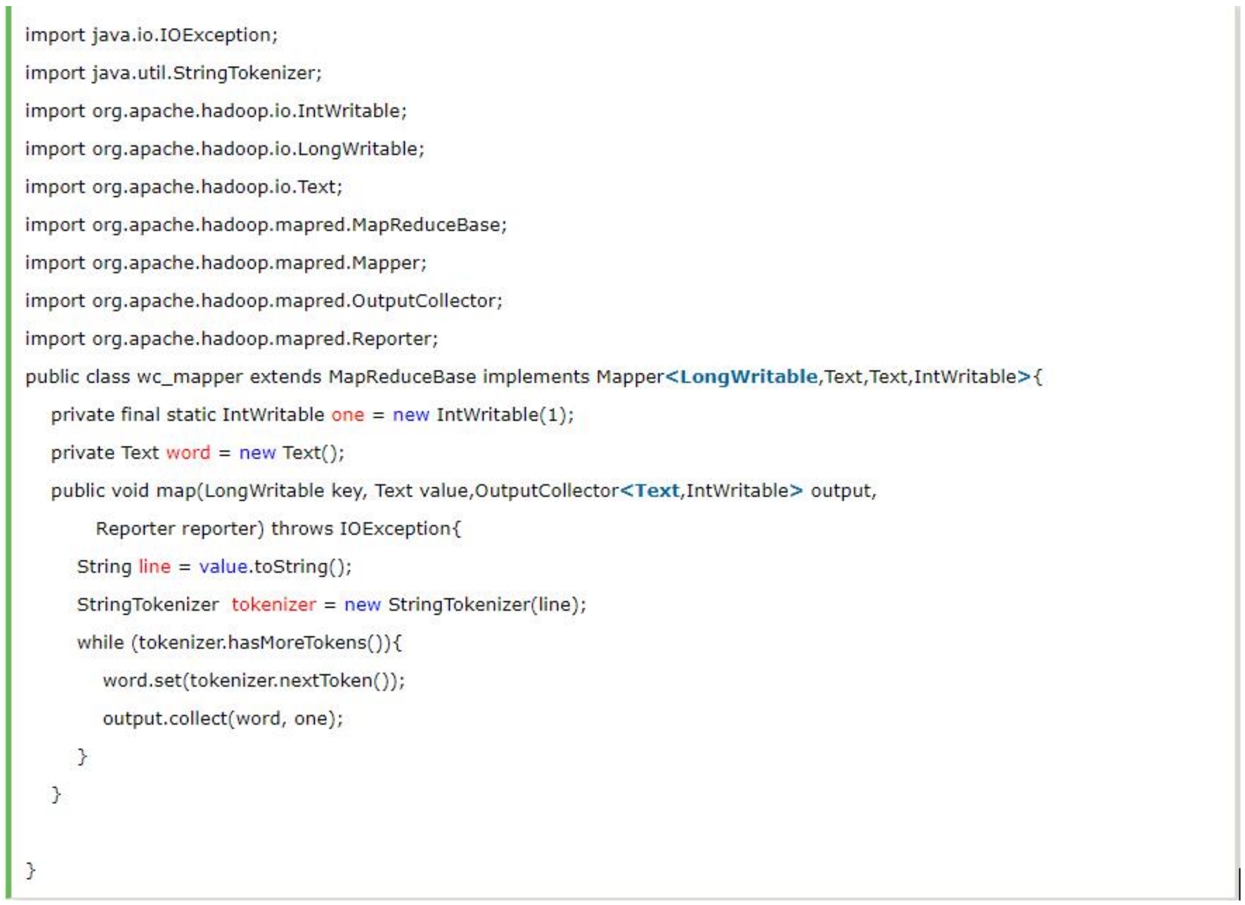
Solution:

Step 1: Upload the file on HDFS data.txt from /usr/Desktop(local path) to /Hadoop/data (Hadoop folder).

$hadoop fs ?put /usr/Desktop/data.txt /Hadoop/data

Step 2: Write the Map reduce program using eclipse and make the jar of it and name it count.

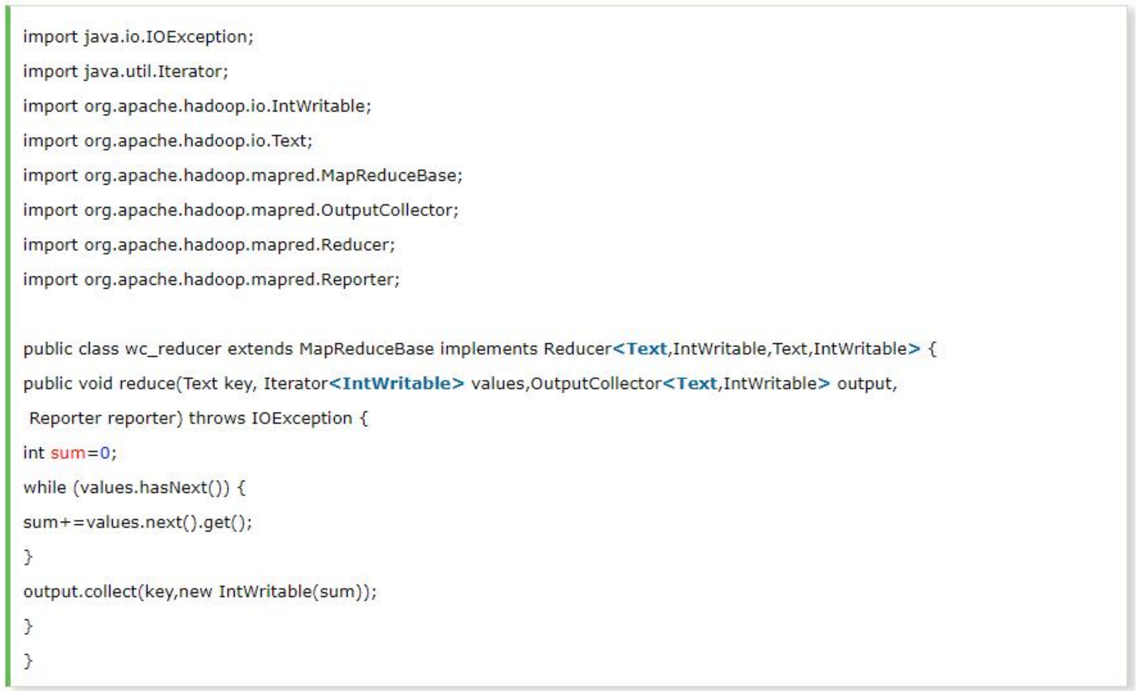
File: wc_mapper.java



```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class wc_mapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,
           Reporter reporter) throws IOException{
        String line = value.toString();
        StringTokenizer  tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()){
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }


}
```

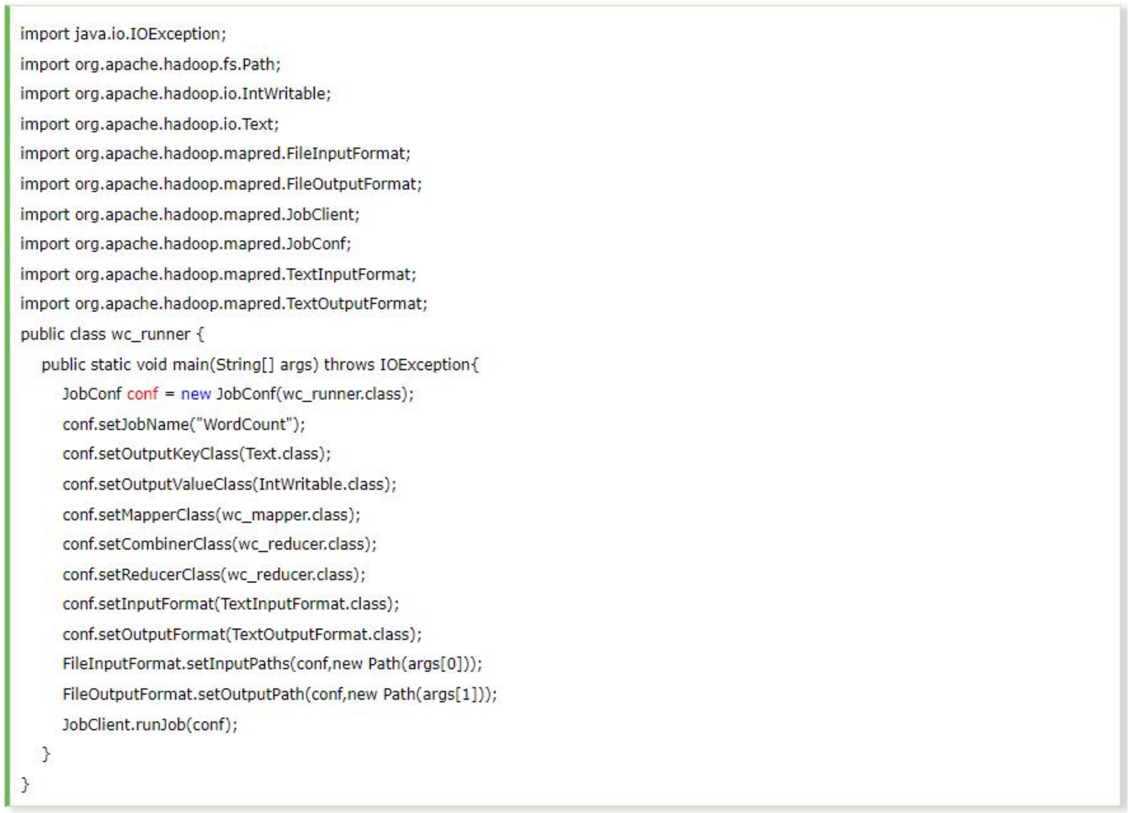
File: wc_reducer.java



```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class wc_reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable> {
public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,
 Reporter reporter) throws IOException {
int sum=0;
while (values.hasNext()) {
sum+=values.next().get();
}
output.collect(key,new IntWritable(sum));
}
}
```

File: wc_runner.java

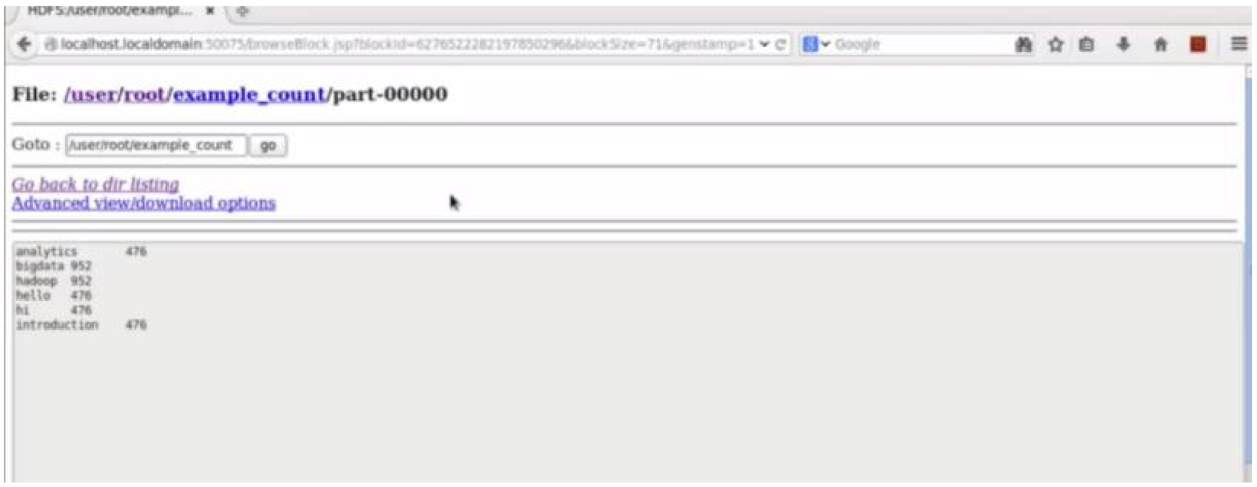


```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
public class wc_runner {
   public static void main(String[] args) throws IOException{
      JobConf conf = new JobConf(wc_runner.class);
      conf.setJobName("WordCount");
      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);
      conf.setMapperClass(wc_mapper.class);
      conf.setCombinerClass(wc_reducer.class);
      conf.setReducerClass(wc_reducer.class);
      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);
      FileInputFormat.setInputPaths(conf,new Path(args[0]));
      FileOutputFormat.setOutputPath(conf,new Path(args[1]));
      JobClient.runJob(conf);
   }
}
```

Step 3: Run the jar file

```
$hadoop jar count.jar WordCount /Hadoop/data.txt/user/root/example_count
```

The output is stored in example_countfolder.



File: /user/root/example_count/part-00000

Goto : /user/root/example_count go

Go back to dir listing
Advanced view/download options

```
analytics	476
bigdata 952
hadoop	952
hello	476
hi	476
introduction	476
```