CHAPTER 1: BIG DATA

1.1 What is Big Data?

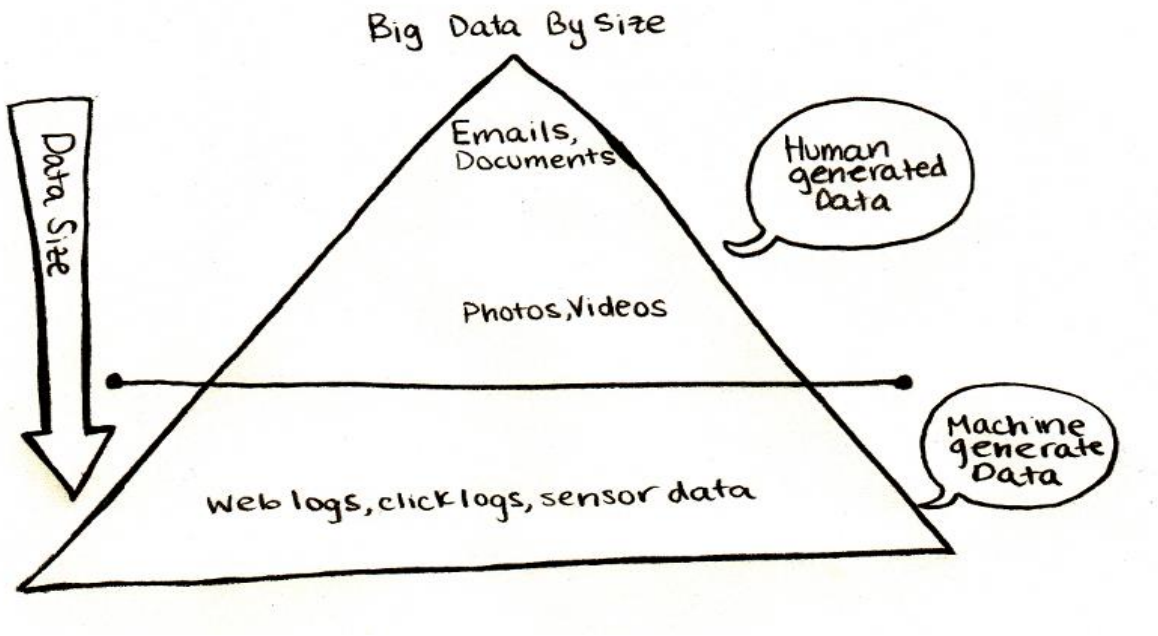
Big Data is very large, loosely structured data set that defies traditional storage.

1.2 Human Generated Data and Machine Generated Data

Human Generated Data is emails, documents, photos and tweets. We are generating this data faster than ever. Just imagine the number of videos uploaded to You Tube and tweets swirling around. This data can
be Big Data too.

Machine Generated Data is a new breed of data. This category consists of sensor data, and logs generated by 'machines' such as email logs, click stream logs, etc. Machine generated data is orders of magnitude larger than Human Generated Data.

Before 'Hadoop' was in the scene, the machine generated data was mostly ignored and not captured. It is because dealing with the volume was NOT possible, or NOT cost effective.



1.3 Where does Big Data come from

Original big data was the web data -- as in the entire Internet! Remember Hadoop was built to index the web. These days Big data comes from multiple sources.

• **Web Data** -- Still it is big data.

• **Social media data** : Sites like Facebook, Twitter, LinkedIn generate a large amount of data.

• **Click stream data** : When users navigate a website, the clicks are logged for further analysis (like navigation patterns). Click stream data is important in on line advertising and and E-Commerce.

• **Sensor data** : Sensors embedded in roads to monitor traffic and misc. other applications generate a large volume of data.

• **Connected Devices** : Smart phones are a great example. For example when you use a navigation application like Google Maps or Waze, your phone sends pings back reporting its location and speed (this information is used for calculating traffic hotspots). Just imagine hundres of millions (or even billions)
of devices consuming data and generating data.

1.4 Examples of Big Data in the Real world

So how much data are we talking about?

• **Facebook** : Has 40 PB of data and captures 100 TB / day

• **Yahoo** : 60 PB of data

• **Twitter** : 8 TB / day

• **EBay** : 40 PB of data, captures 50 TB / day

1.5 Challenges of Big Data

Sheer size of Big Data

Big data is... well... big in size! How much data constitute Big Data is not very clear cut. So lets not get bogged down in that debate. For a small company that is used to dealing with data in gigabytes, 10TB of data would be BIG. However for companies like Facebook and Yahoo, peta bytes is big.

Just the size of big data, makes it impossible (or at least cost prohibitive) to store in traditional storage like databases or conventional filers.
We are talking about cost to store gigabytes of data. Using traditional storage filers can cost a lot of money to store Big Data.

Big Data is unstructured or semi structured

A lot of Big Data is unstructured. For example click stream log data might look like time stamp,user_id, page, referrer_page.

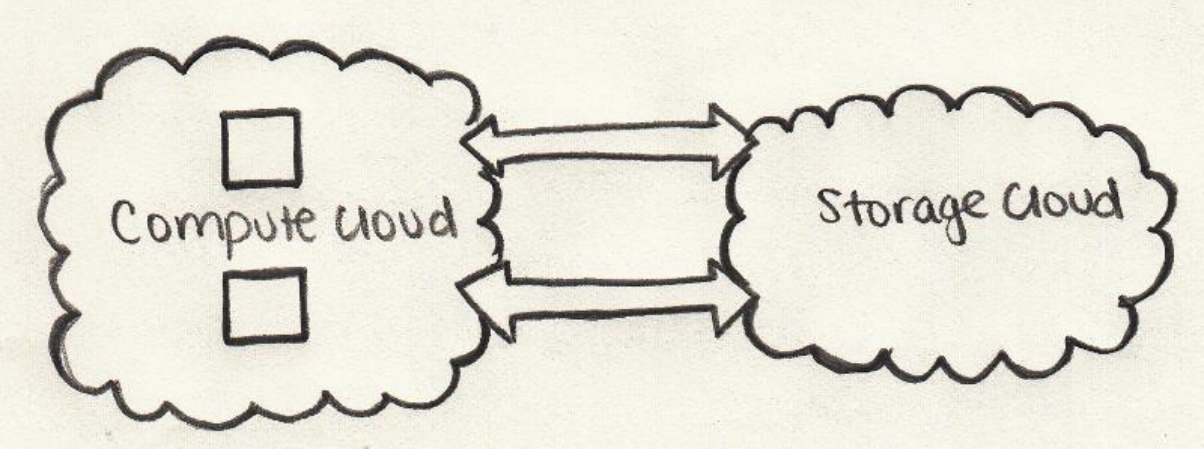
Lack of structure makes relational databases not well suited to store Big Data.

Plus, not many databases can cope with storing billions of rows of data.

No point in just storing big data, if we can't process it

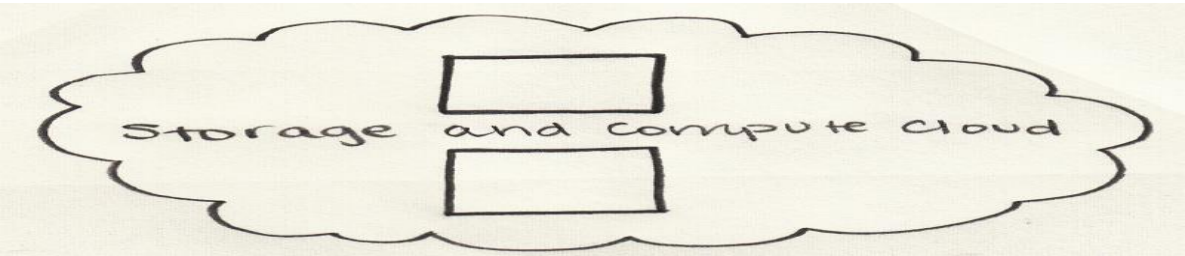
Storing Big Data is part of the game. We have to process it to mine intelligence out of it. Traditional storage systems are pretty 'dumb' as in they just store bits -- They don't offer any processing power.

The traditional data processing model has data stored in a 'storage cluster', which is copied over to a 'compute cluster' for processing, and the results are written back to the storage cluster.



This model however doesn't quite work for Big Data because copying so much data out to a compute cluster might be too time consuming or impossible. So what is the answer?

One solution is to process Big Data 'in place' -- as in a storage cluster doubling as a compute cluster.



CHAPTER 2: BIG DATA AND HADOOP

What is Hadoop?

Hadoop is an open source software stack that runs on a cluster of machines. Hadoop provides distributed storage and distributed processing for very large data sets

What is the license of Hadoop?

Hadoop is open source software. It is an Apache project released under Apache Open Source License v2.0. This license is very commercial friendly.

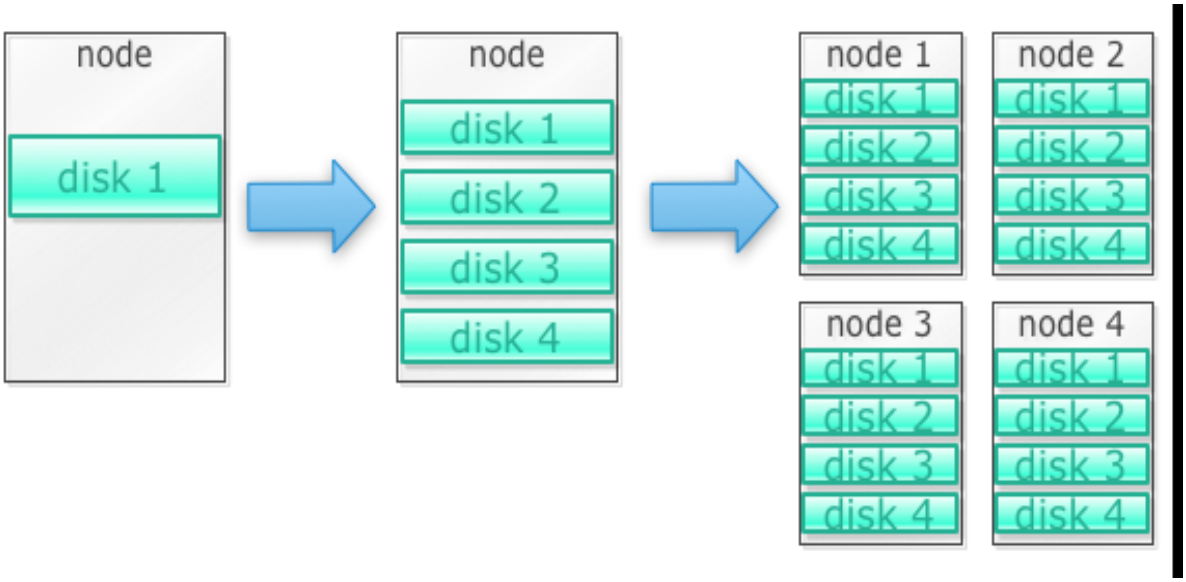
Who contributes to Hadoop?

Originally Hadoop was developed and open sourced by Yahoo. Now Hadoop is developed as an Apache Software Foundation project and has numerous contributors from Cloudera, Horton Works, Facebook, etc.

2.1 How Hadoop solves the Big Data problem

Hadoop is built to run on a cluster of machines

Lets start with an example. Say we need to store lots of photos. We will start with a single disk. When we exceed a single disk, we may use a few disks stacked on a machine. When we max out all the disks on a single machine, we need to get a bunch of machines, each with a bunch of disks.



This is exactly how Hadoop is built. Hadoop is designed to run on a cluster of machines from the get go

Hadoop clusters scale horizontally

More storage and compute power can be achieved by adding more nodes to a Hadoop cluster. This eliminates the need to buy more and more powerful and expensive hardware.

Hadoop can handle unstructured / semi-structured data

Hadoop doesn't enforce a 'schema' on the data it stores. It can handle arbitrary text and binary data. So Hadoop can 'digest' any unstructured data easily.

Hadoop clusters provides storage and computing

We saw how having separate storage and processing clusters is not the best fit for Big Data. Hadoop clusters provide storage and distributed computing all in one.

2.2 Business Case for Hadoop

Hadoop provides storage for Big Data at reasonable cost

Storing Big Data using traditional storage can be expensive. Hadoop is built around commodity hardware.

Hence it can provide fairly large storage for a reasonable cost. Hadoop has been used in the field at Peta byte scale.

One study by Cloudera suggested that Enterprises usually spend around $25,000 to $50,000 dollars per tera byte per year. With Hadoop this cost drops to few thousands of dollars per tera byte per year. And hardware gets cheaper and cheaper this cost continues to drop.

Hadoop allows to capture new or more data

Some times organizations don't capture a type of data, because it was too cost prohibitive to store it. Since Hadoop provides storage at reasonable cost, this type of data can be captured and stored.

One example would be web site click logs. Because the volume of these logs can be very high, not many organizations captured these. Now with Hadoop it is possible to capture and store the logs

With Hadoop, you can store data longer

To manage the volume of data stored, companies periodically purge older data. For example only logs for the last 3 months could be stored and older logs were deleted. With Hadoop it is possible to store the historical data longer. This allows new analytics to be done on older historical data.

For example, take click logs from a web site. Few years ago, these logs were stored for a brief period of time to calculate statics like popular pages ..etc. Now with Hadoop it is viable to store these click logs for longer period of time.

Hadoop provides scalable analytics

There is no point in storing all the data, if we can't analyze them. Hadoop not only provides distributed storage, but also distributed processing as well. Meaning we can crunch a large volume of data in parallel.

The compute framework of Hadoop is called Map Reduce. Map Reduce has been proven to the scale of peta bytes

Hadoop provides rich analytics

Native Map Reduce supports Java as primary programming language. Other languages like Ruby, Python and R can be used as well.

Of course writing custom Map Reduce code is not the only way to analyze data in Hadoop. Higher level Map Reduce is available. For example a tool named Pig takes english like data flow language and translates them into Map Reduce. Another tool Hive, takes SQL queries and runs them using Map Reduce

Business Intelligence (BI) tools can provide even higher level of analysis. Quite a few BI tools can work with Hadoop and analyze data stored in Hadoop. For a list of BI tools that support Hadoop .

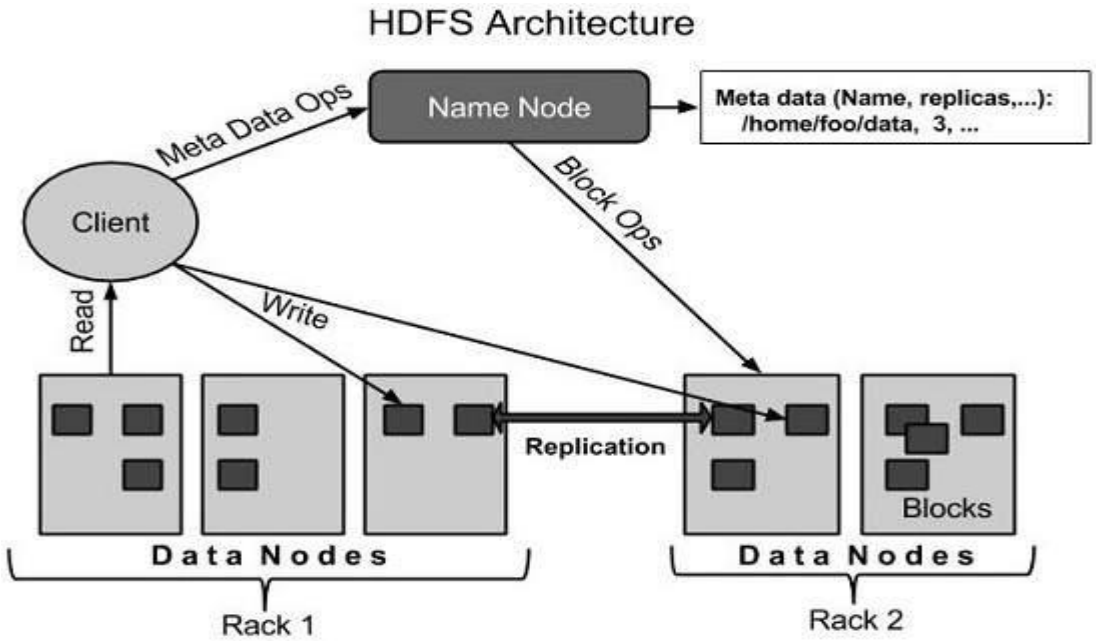
CHAPTER 3: HADOOP DISTRIBUTED FILE SYSTEM(HDFS)

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

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

Features of HDFS

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



3.1 Namenode

The Namenode server in HDFS is the most important part, it store all the metadata of the stored files and directories, such as the list of files, list of blocks for each file, list of DataNodes for each block and also File attributes. The other role is to serve the client queries, it allows clients to add/copy/move/delete a file, it will records the actions into a transaction log. For the performance, it save the whole file structure tree in RAM and hard drive. A HDFS only allow one running namnode, that's why it is a single point of failure, if the namenode failed or goes down, the whole file system will goes offline too. So, for the namenode machine, we need to take special cares on it, such as adding more RAM to it, this will increase the file system capacity, and do not make it as DataNode, JobTracker and other optional roles

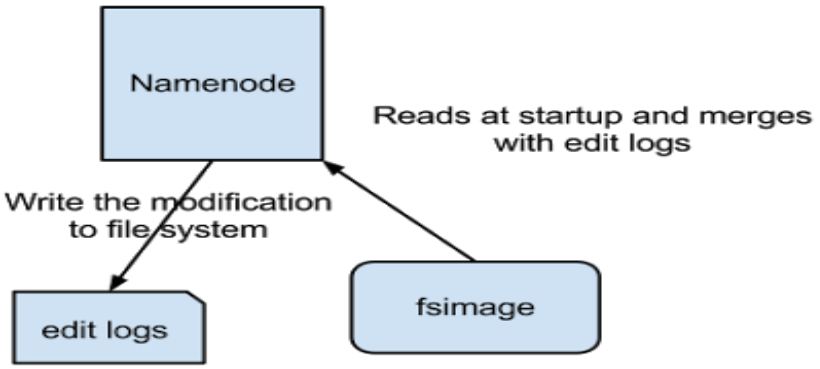
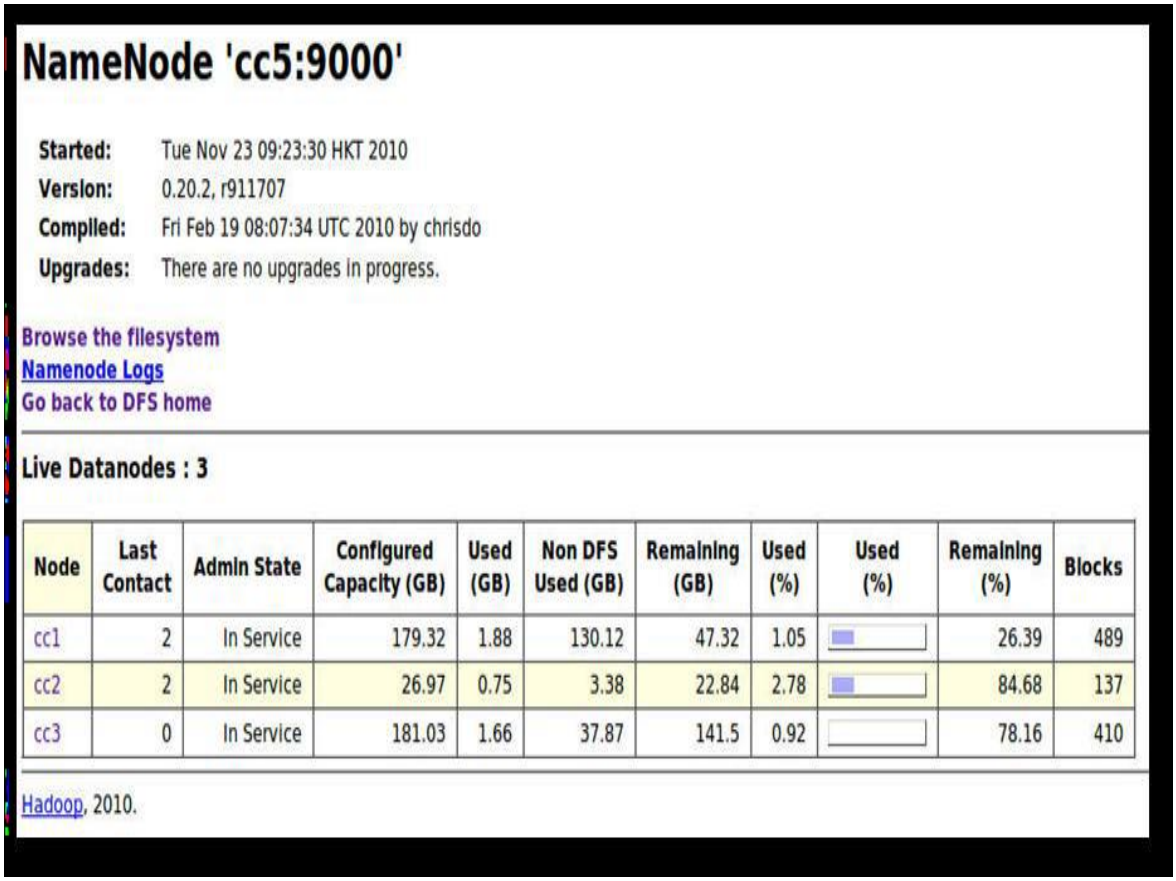


Fig show how Name Node stores information in disk.



NameNode 'cc5:9000'

Started: Tue Nov 23 09:23:30 HKT 2010
Version: 0.20.2, r911707
Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo
Upgrades: There are no upgrades in progress.

Browse the filesystem
Namenode Logs
Go back to DFS home

Live Datanodes : 3

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
cc1	2	In Service	179.32	1.88	130.12	47.32	1.05		26.39	489
cc2	2	In Service	26.97	0.75	3.38	22.84	2.78		84.68	137
cc3	0	In Service	181.03	1.66	37.87	141.5	0.92		78.16	410

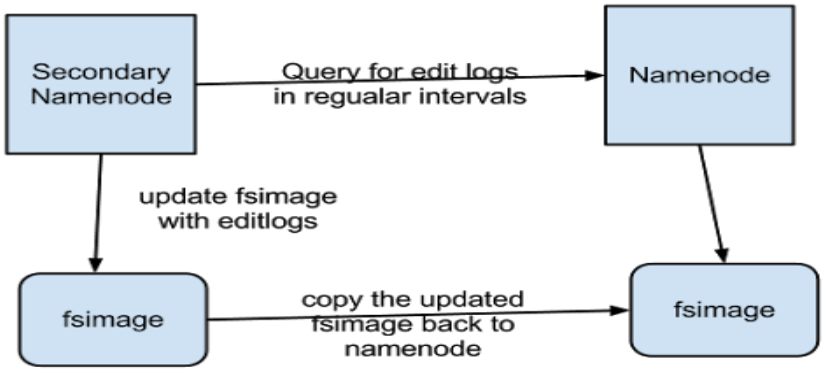
Hadoop, 2010.

Fig web interface of namenode

3.2 Secondary Namenode

From the name, it may like the active-standby node of the NameNode, in fact, it is not. What it works is to backup the metadata and store it to the hard disk, this may helping to reduce the restarting time of NameNode. In HDFS, the recent actions on HDFS will be stored in to a file called EditLog on the NameNode, after restarting HDFS; the NameNode will replay according to the Editlog. SecondaryNameNode will periodically combines the content of EditLog into a checkpoint and clear the Editlog File, after that, the NameNode will replay start from the latest checkpoint, the restarting time of NameNode will be reduced.

SecondaryNamede was running on the same machine as NameNode in default which is not a good idea, it is because if the NameNode machine crashed, it will hard to restore. Placing the SecondaryNamede on the other machine, it may help the new NameNode to restore the file structure.

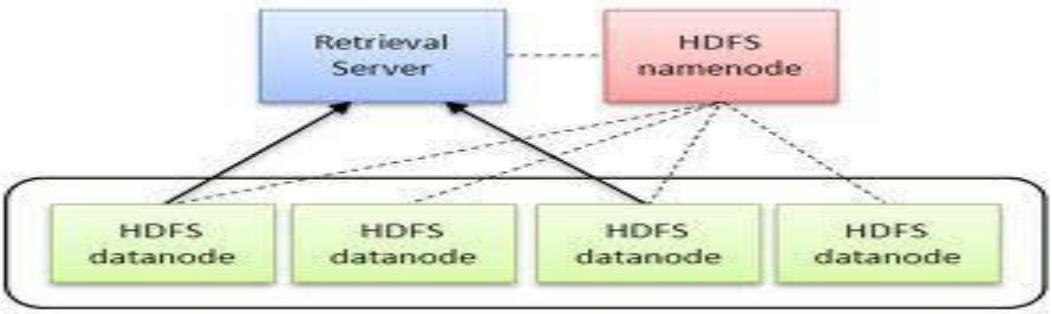


3.3 Datanode

Datanode is the scaled up part of HDFS, it may have thousands, it mainly use to store the file data. On startup, DataNode will connect to the NameNode and get ready to respond to the operations from NameNode. After the NameNode telling the position of a file to the client, the client will directly talk to the DataNode to access the files. DataNodes could also talk to each other when they replicating data. The DataNode will also periodically send a report of all existing blocks to the NameNode and validates the data block checksums (CRC32).

Objective of Datanode

- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.



3.4 Other Features

HeartBeats

DataNodes send heartbeat to the NameNode, and NameNode used heartbeats to detect DataNode failure. If failure detected, the NameNode will choose New DataNodes for new replicas to balances disk usage and communication traffic to DataNodes.

Rebalancer

The rebalance preferred similar percentage of disk usage of each DataNode. It works when DataNodes are added or removed. It is throttled to avoid network congestion.

Block Placement

Any single file stored in Hadoop will be stored in different machines, for the metadata, it will be stored on NameNode, the file content will be split to data blocks and store each block to different DataNodes, the block side is 64MB in default, and it will has 3 replicas, the first one stored on local node, the 2nd and 3rd replicas are stored on remote nodes, for the other replicates, it will be placed randomly. When the client read a file, it will be directed to the nearest replicate.

Data Correctness

In HDFS, if use CRC32 to do the checksum validation, if validation failed, the client will try the other replicas.

3.5 Compare with NFS

Network File System(NFS) is one of the famous DFS, the design is straightforward, it provides remote access to single logical volume stored on a single machine, but there has limitations such as the files in an NFS volume all reside on a single machine, lots of information as can be stored in one machine, and does not provide any reliability guarantees if that machine goes down, another limitation is that all the data is stored on a single machine, all the clients must go to this machine to retrieve their data. This can overload the server if a large number of clients must be handled. HDFS solve the problems

HDFS should integrate well with Hadoop MapReduce, allowing data to be read and computed upon locally when possible.

Applications that use HDFS are assumed to perform long sequential streaming reads from files. HDFS is optimize HDFS is designed to store a very large amount of information (terabytes or petabytes). This requires spreading the data across a large number of machines. It also supports much larger file sizes than NFS.

HDFS should store data reliably. If individual machines in the cluster malfunction, data should still be available.

HDFS should provide fast, scalable access to this information. It should be possible to serve a larger number of clients by simply adding more machines to the cluster.

To provide streaming read performance; this comes at the expense of random seek times to arbitrary positions in files.

Data will be written to the HDFS once and then read several times; updates to files after they have already been closed are not supported. (An extension to Hadoop will provide support for appending new data to the ends of files; it is scheduled to be included in Hadoop 0.19 but is not available yet.)

Due to the large size of files, and the sequential nature of reads, the system does not provide a mechanism for local caching of data. The overhead of caching is great enough that data should simply be re-read from HDFS source.

Individual machines are assumed to fail on a frequent basis, both permanently and intermittently. The cluster must be able to withstand the complete failure of several machines, possibly many happening at the same time (e.g., if a rack fails all together). While performance may degrade proportional to the number of machines lost, the system as a whole should not become overly slow, nor should information be lost. Data replication strategies combat this problem.

3.6 Limitation of HDFS

MapReduce/ Hadoop File System lands data between reduce steps, which is a huge performance constraint.

Hadoop File System centrally manages an index necessary to map tasks to the data distributed throughout the nodes— a documented bottleneck.

Operations requiring data collocation to compute the result (joins, aggregations, sorts, deduplications and so on) will run inefficiently when the data distribution is different from the index.

Hadoop is not a good choice when real-t ime, low-latency processing is required because there is no “real-time” version of Hadoop available.

Job start- up is slow, which can be a big performance penalty, particularly for small, bursty jobs

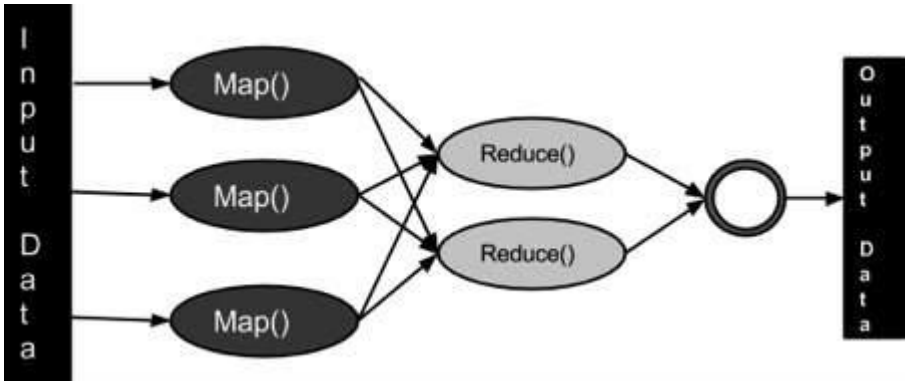
CHAPTER 4:MAPREDUCE

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job

The Algorithm

- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



There has 2 roles for MapReduce jobs, one is JobTracker and TaskTracker in hadoop version1.

4.1 Job Tracker

- JobTracker process runs on a separate node and not usually on a DataNode.

- JobTracker is an essential Daemon for MapReduce execution in MRv1. It is replaced by ResourceManager/ApplicationMaster in MRv2.
- JobTracker receives the requests for MapReduce execution from the client.
- JobTracker talks to the NameNode to determine the location of the data.
- JobTracker finds the best TaskTracker nodes to execute tasks based on the data locality (proximity of the data) and the available slots to execute a task on a given node.
- JobTracker monitors the individual TaskTrackers and the submits back the overall status of the job back to the client.
- JobTracker process is critical to the Hadoop cluster in terms of MapReduce execution.

When the JobTracker is down, HDFS will still be functional but the MapReduce execution can not be started and the existing MapReduce jobs will be halted.

4.2 TaskTracker

- TaskTracker runs on DataNode. Mostly on all DataNodes.
- TaskTracker is replaced by Node Manager in MRv2.
- Mapper and Reducer tasks are executed on DataNodes administered by TaskTrackers.
- TaskTrackers will be assigned Mapper and Reducer tasks to execute by JobTracker.
- TaskTracker will be in constant communication with the JobTracker signalling the progress of the task in execution.
- TaskTracker failure is not considered fatal. When a TaskTracker becomes unresponsive, JobTracker will assign the task executed by the TaskTracker to another node.

4.3 Node Manager

Node Manager is a Java utility that runs as separate process from WebLogic Server and allows you to perform common operations tasks for a Managed Server, regardless of its location with respect to its Administration Server. While use of Node Manager is optional, it provides valuable benefits if your WebLogic Server environment hosts applications with high availability requirements.

If you run Node Manager on a machine that hosts Managed Servers, you can start and stop the Managed Servers remotely using the Administration Console or from the command line. Node Manager can also automatically restart a Managed Server after an unexpected failure.

4.4 ResourceManager

The fundamental idea of YARN is to split up the functionalities of resource management and job scheduling/monitoring into separate daemons. The idea is to have a global ResourceManager (*RM*) and per-application ApplicationMaster (*AM*). An application is either a single job or a DAG of jobs.

The ResourceManager and the NodeManager form the data-computation framework. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The NodeManager is the per-machine framework agent who is responsible for containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager/Scheduler.

The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks

4.5 Features

Automatic parallelization and distribution

When we run our application, the JobTracker will automatically handle all the messy things, distribute tasks, failure handling, report progress.

Fault-tolerance

The TaskTracker nodes are monitored. If they do not submit heartbeat signals in a period of time, they are deemed to have failed and the work is scheduled on a different TaskTracker. If a task failed for 4 times (in default), the whole job will failed.
A TaskTracker will notify the JobTracker when a task fails. The JobTracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may even blacklist the TaskTracker as unreliable.

Locality optimization

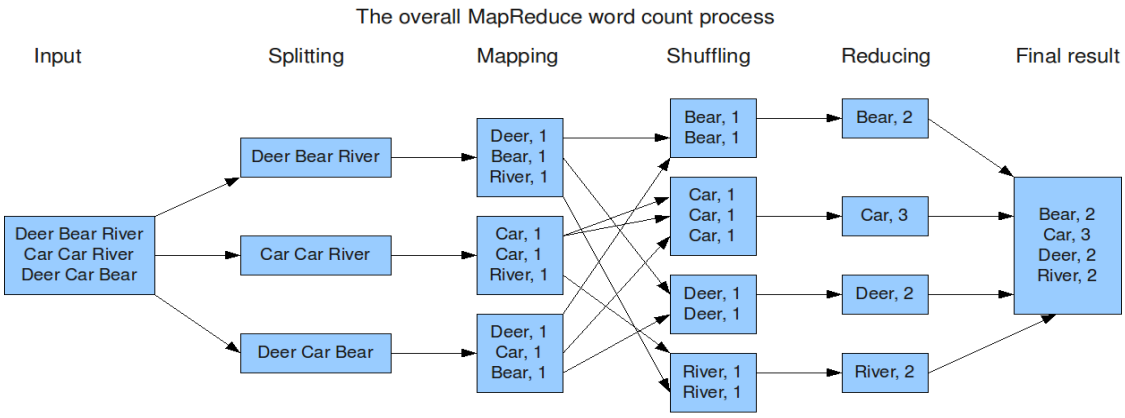
He JobTracker will assign the splits which have the same keys to nearest clusters; this may reduce the job distribution time in Map procedure and collection the output key-value pairs in Reduce procedure

Task Attempts	Machine	Status	Progress	Start Time	Finish Time	Errors
attempt_201011232109_0001_m_000003_0	Task attempt: /default-rack/cc3 Cleanup Attempt: /default-rack/cc3	FAILED	100.00%	24-Nov-2010 04:55:28	24-Nov-2010 05:05:50 (10mins, 21sec)	Task attempt_201011232109_
attempt_201011232109_0001_m_000003_1	/default-rack/cc2	RUNNING	0.00%	23-Nov-2010 20:55:54		
attempt_201011232109_0001_m_000003_2	/default-rack/cc1	RUNNING	0.00%	23-Nov-2010 21:32:48		

Backup Tasks

In MapReduce, some operations may be come straggler and increase the total running time, one of the most important reason is that some of the Mapper/Reducer will fight for the local resources such as CPU, Memory, local disk, and network bandwidth, etc. these may increase the delay latency. A simple solution in Hadoop is to have more than 1 copies of the Map/Reduce job running in different machines if there has some empty resources/slots. If a copy finished the job, the others will be killed. The overhead of this procedure is small but the total time reduced significantly in large scale clusters operations

Example Of MapReduce WordCount Process



CHAPTER 5: HADOOP ENVIRONMENT SETUP

Step 1: Setting Up Hadoop

You can set Hadoop environment variables by appending the following commands to **~/.bashrc** file.

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_INSTALL=$HADOOP_HOME
```

Step 2: Hadoop Configuration

You can find all the Hadoop configuration files in the location "$HADOOP_HOME/etc/hadoop". It is required to make changes in those configuration files according to your Hadoop infrastructure.

```
$ cd $HADOOP_HOME/etc/hadoop
```

In order to develop Hadoop programs in java, you have to reset the java environment variables in **hadoop-env.sh** file by **JAVA_HOME** value with the location of java in your system.

```
export JAVA_HOME=/usr/local/jdk1.7.0_71
```

The following are the list of files that you have to edit to configure Hadoop.

core-site.xml

The **core-site.xml** file contains information such as the port number used for Hadoop instance, memory allocated for the file system, memory limit for storing the data, and size of Read/Write buffers.

Open the core-site.xml and add the following properties in between <configuration>, </configuration> tags.

```
<configuration>

  <property>
    <name>fs.default.name </name>
    <value> hdfs://localhost:9000 </value>
  </property>

</configuration>
```

hdfs-site.xml

The **hdfs-site.xml** file contains information such as the value of replication data, namenode path, and datanode paths of your local file systems. It means the place where you want to store the Hadoop infrastructure.

```
<configuration>

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

  <property>
    <name>dfs.name.dir</name>
    <value>file:///home/hadoop/hadoopinfra/hdfs/namenode </value>
  </property>

  <property>
    <name>dfs.data.dir</name>
    <value>file:///home/hadoop/hadoopinfra/hdfs/datanode </value>
  </property>
```

```
</configuration>
```

yarn-site.xml

This file is used to configure yarn into Hadoop. Open the yarn-site.xml file and add the following properties in between the <configuration>, </configuration> tags in this file.

```
<configuration>

  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>

</configuration>
```

mapred-site.xml

This file is used to specify which MapReduce framework we are using. By default, Hadoop contains a template of yarn-site.xml. First of all, it is required to copy the file from **mapred-site,xml.template** to **mapred-site.xml** file using the following command.

```
$ cp mapred-site.xml.template mapred-site.xml
```

Open mapred-site.xml file and add the following properties in between the <configuration>, </configuration>tags in this file.

```
<configuration>

  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>

</configuration>
```

Verifying Hadoop Installation

The following steps are used to verify the Hadoop installation.

Step 1: Name Node Setup

Set up the namenode using the command "hdfs namenode -format" as follows.

```
$ cd ~
$ hdfs namenode -format
```

The expected result is as follows.

```
10/24/14 21:30:55 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost/192.168.1.11
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.4.1
...
...
10/24/14 21:30:56 INFO common.Storage: Storage directory
/home/hadoop/hadoopinfra/hdfs/namenode has been successfully formatted.
10/24/14 21:30:56 INFO namenode.NNStorageRetentionManager: Going to
retain 1 images with txid >= 0
10/24/14 21:30:56 INFO util.ExitUtil: Exiting with status 0
10/24/14 21:30:56 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at localhost/192.168.1.11
************************************************************/
```

Step 2: Verifying Hadoop dfs

The following command is used to start dfs. Executing this command will start your Hadoop file system.

```
$ start-dfs.sh
```

The expected output is as follows:

```
10/24/14 21:37:56
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/hadoop/hadoop
2.4.1/logs/hadoop-hadoop-namenode-localhost.out
localhost: starting datanode, logging to /home/hadoop/hadoop
2.4.1/logs/hadoop-hadoop-datanode-localhost.out
Starting secondary namenodes [0.0.0.0]
```

Step 3: Verifying Yarn Script

The following command is used to start the yarn script. Executing this command will start your yarn daemons.

```
$ start-yarn.sh
```

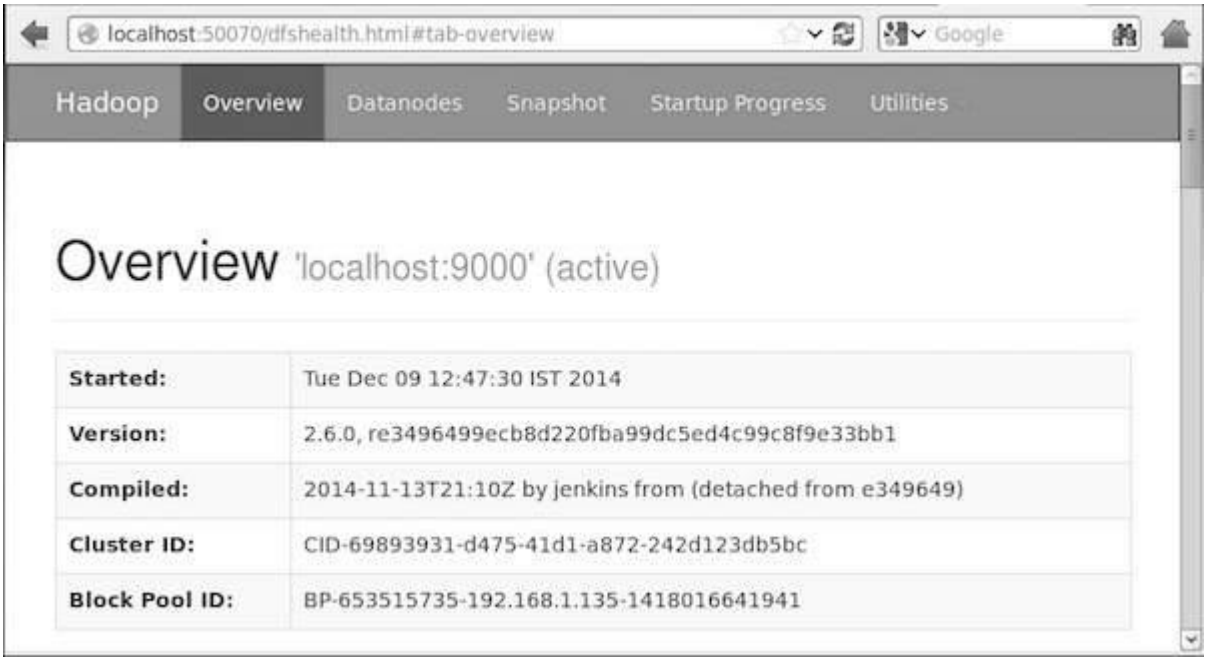
The expected output as follows:

```
starting yarn daemons
starting resourcemanager, logging to /home/hadoop/hadoop
2.4.1/logs/yarn-hadoop-resourcemanager-localhost.out
localhost: starting nodemanager, logging to /home/hadoop/hadoop
2.4.1/logs/yarn-hadoop-nodemanager-localhost.out
```

Step 4: Accessing Hadoop on Browser

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

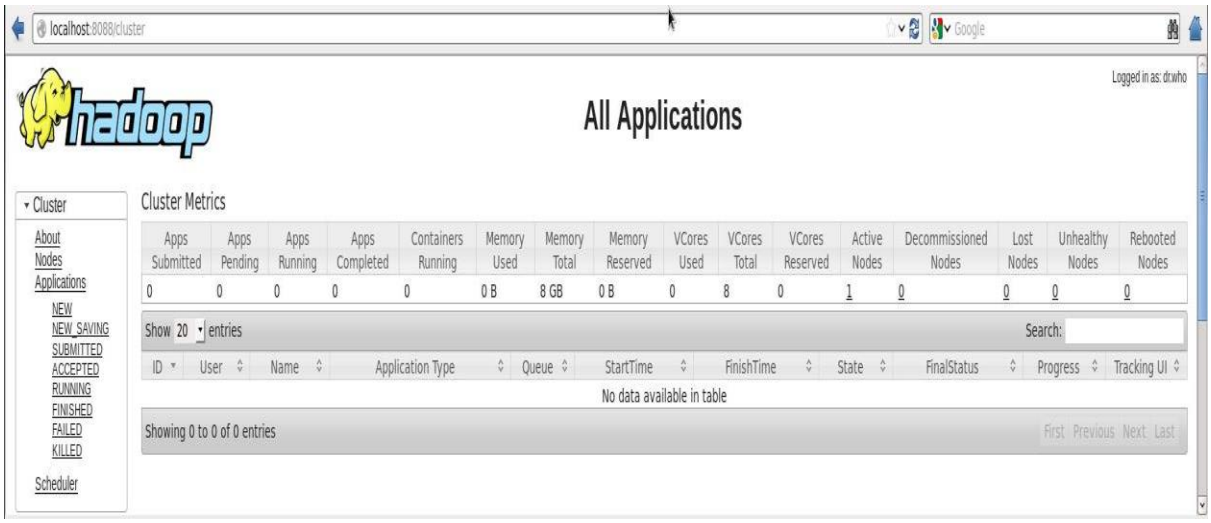
```
http://localhost:50070/
```



Step 5: Verify All Applications for Cluster

The default port number to access all applications of cluster is 8088. Use the following url to visit this service.

```
http://localhost:8088/
```



CONCLUSION

During my 60 days practical training at LINUX WORLD,JAIPUR.I got information and experience from my relevant section.

Technology:Hadoop

In big data using hadoop framework firstly I learnt about what is big data, big data comes from,challenges of big data,how to solve big data isssue,solve using hadoop framework,understand its architecture ,methology and its implementation details and also learn web technology that is used in to create a web interface of our hadoop cluster. After that I learnt that how to configure hadoop cluster.after making simple cluster configure in command mode and I also learnt about various way to configure hadoop and lastly we learn about how to create hadoop cluster using python cgi.

I am really thankful to **Mr**. **Vimal Daga**

It was definitely a knowledgeable experience taking training in Jaipur. It is indeed a vast Computer education with lot much to offer us as students

No doubt it showed that mere theoretical and bookish knowledge need to be supplemented with a able practice knowledge. And this very practical knowledge was given very ably by the personals of LINUX WORLD.

LINUX WORLD has helped us in gaining practical knowledge mentioned above to great extent. The staff of LINUX WORLD is very supportive and helpful. They gave their best and in such a easy way in which we can understand. They told us very valuable things which are bard to learn merely from books.

For once again I would like to thank from the core of my heart, to everybody who helped me and cooperated with me for my successful training.