HADOOP CLUSTER

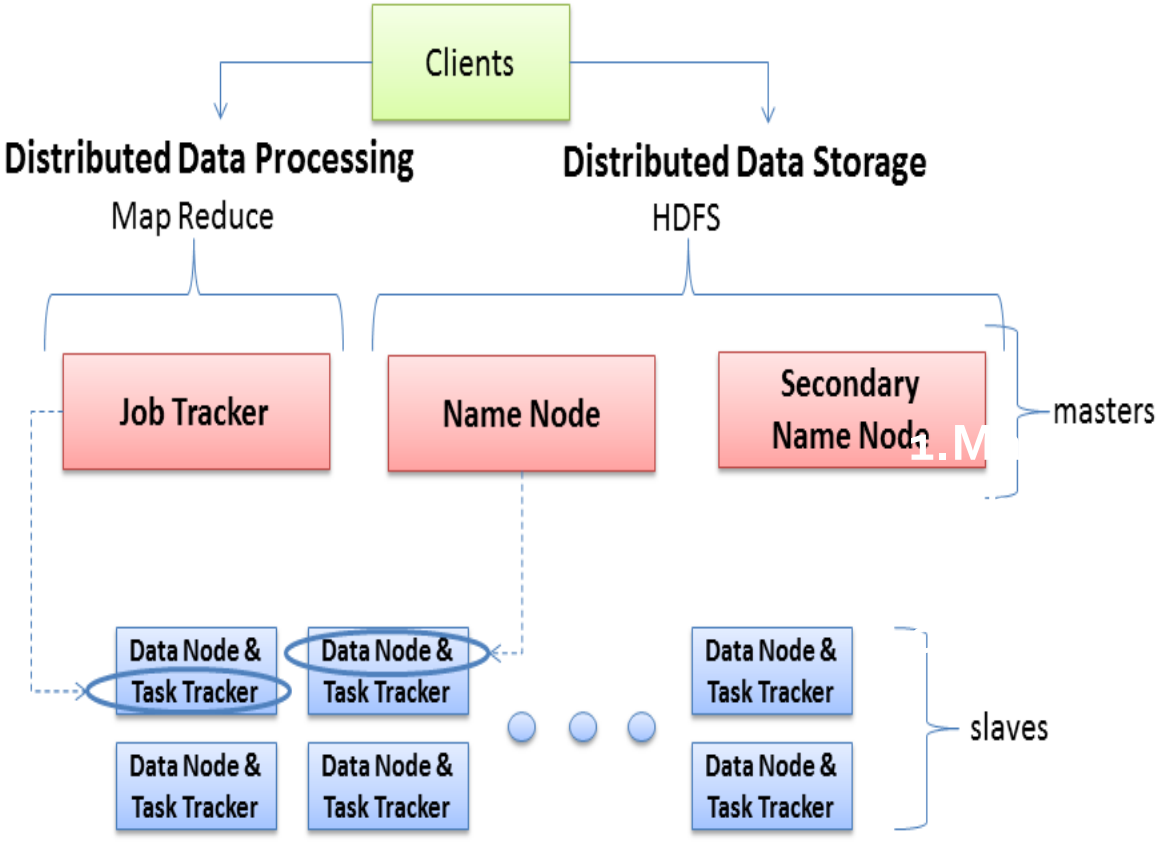
Agenda

- Building blocks of Hadoop cluster
- Few big clusters across the globe
- Cluster Set-up
  - Single Node
- Different components of the Cluster

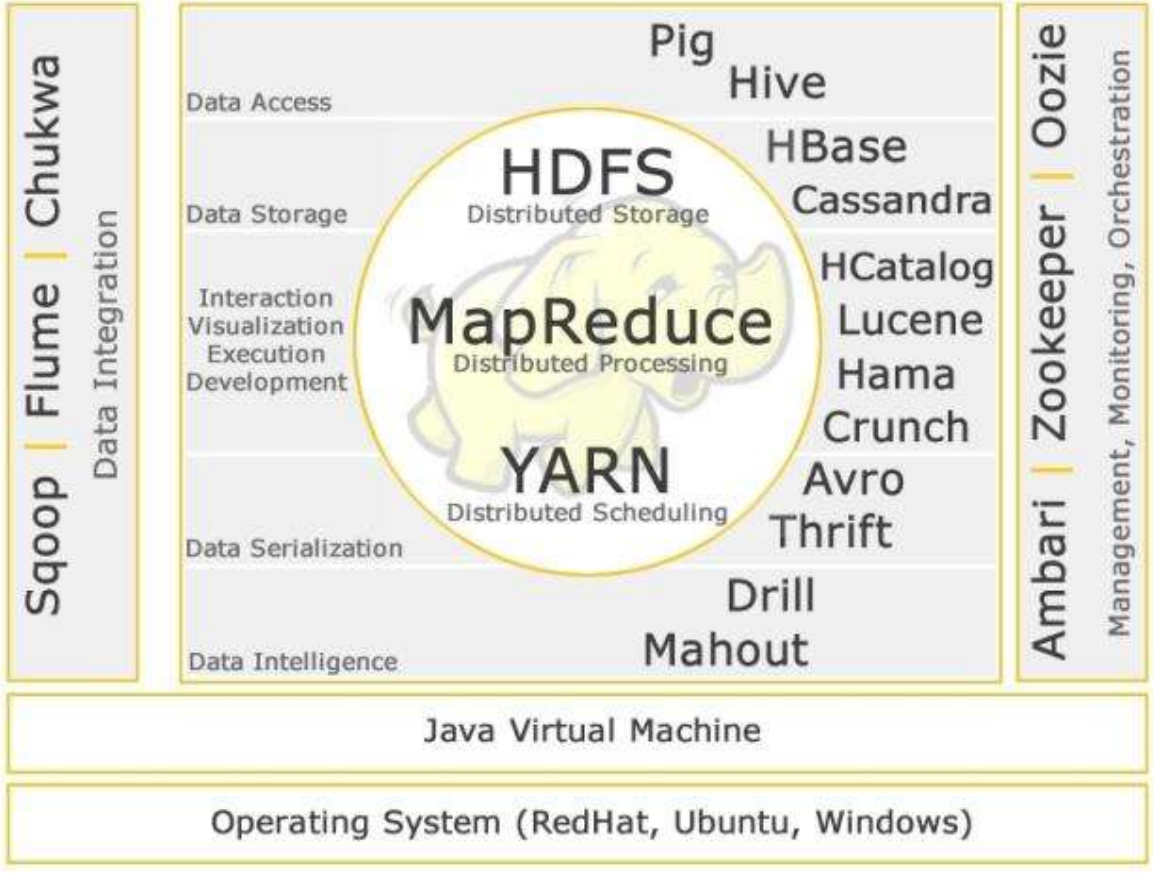
Building blocks

- NameNode
- DataNode
- Secondary NameNode
- Job Tracker
- Task Tracker

Hadoop Server roles

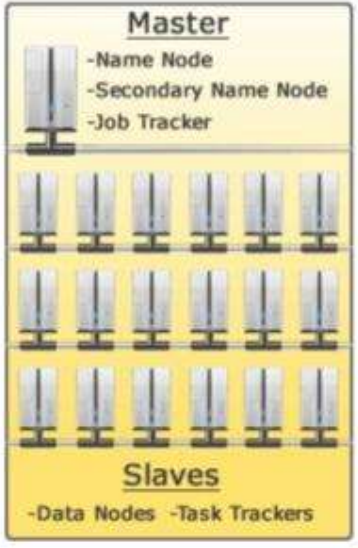


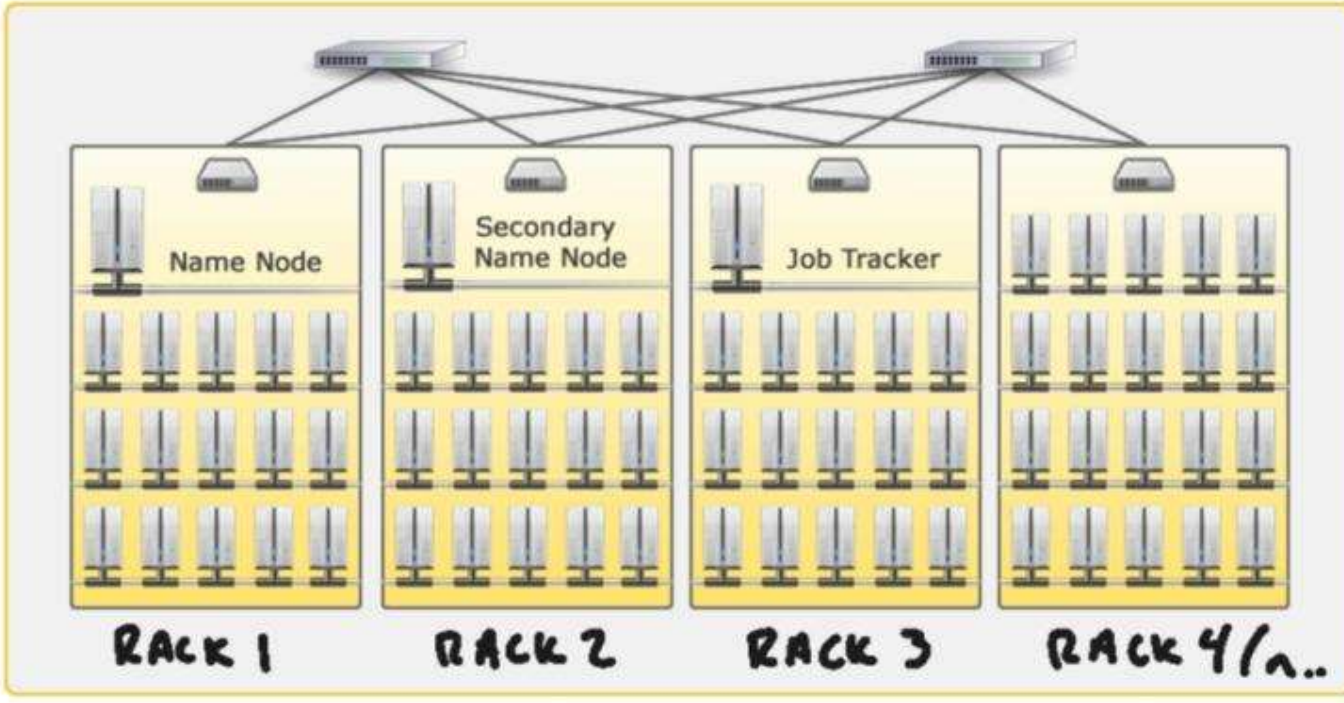
Hadoop Technology Stack



- Hadoop Core
  - HDFS
  - MapReduce
  - YARN (2.0)
- Hadoop Essentials
  - Pig
  - Hive
  - HBase
  - Avro
  - Zookeeper
  - Mahout
  - Oozie
  - Sqoop
- Hadoop Incubator
  - Chuckwa
  - Ambari
  - HDT
  - Hcatalog
  - Knox
  - Spark

HDFS Architecture





NAME NODE = FS OPS, Block mappings

DATA NODE = Block ops, Replication

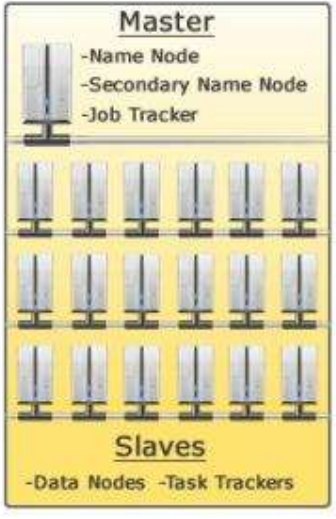
SECONDARY NN = Checkpoint OPS

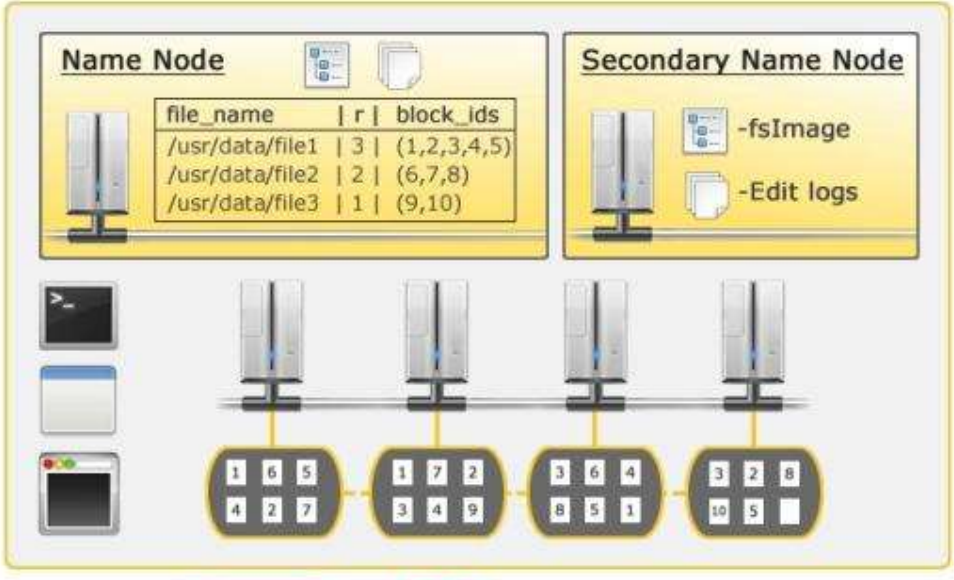
HDFS FEATURES
- Rack awareness
- Reliable storage
- High throughput

HDFS Internals

Data Node

- Handles client requests
- Sends heartbeats to namenodes (3 sec)





Blocks

- 64 MB (default)
- Block placement
- Rebalancing
- Replication Management

✓Name Node

- Contains filesystem metadata
- Clients communicate / controller
- Monitors health

Secondary NameNode

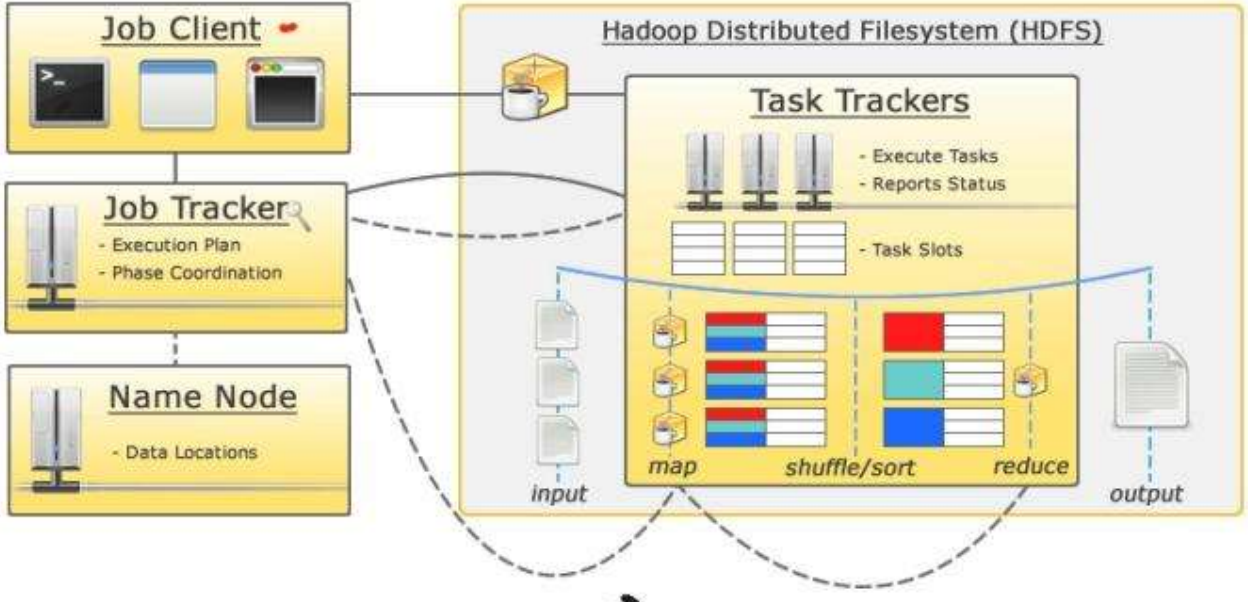
- Housekeeping / Backup of NameNode MD
- Metadata backup can rebuild NameNode
- NOT HIGH AVAILABILITY!

MapReduce Architecture

Job Client
- Submits jobs

Job Tracker
- Coordinates jobs

Task Tracker
- Executes job tasks



1) Client submits job to JobTracker
2) JobTracker talks to NameNode
3) JobTracker creates execution plan
4) JobTracker submits work to TaskTrackers
5) TaskTrackers report progress via heartbeats
6) JobTracker manages phases
7) JobTracker updates status

Name Node

- Distributed storage – HDFS
- NameNode is the master of HDFS
  - Directs Datanodes to perform low level I/O tasks

Name Node

- Keeps track of – how files are broken down and where they are stored.
- This is memory and I/O intensive.
- To reduce the workload, this node doesn't store any data or perform any tasks.

Name Node

- Without name node file system can't be used.
- A single point of failure.
- Name node obliterated → all files lost
- Need to make Name Node resilient enough to withstand failures.
- Constantly engages with data nodes to know their health.

Data Node

- Each slave machine of cluster will be a data node.
- Client request (+)→ Name Node → divide the data into blocks and write in different data nodes.
- Client request (?)→ Name Node → informs client that data is present in so & so node
- Client communicates with datanode.

Data Node

- Datanode may communicate with other datanodes to replicate its blocks.

Data Node

1.Data nodes constantly report to Name Node about the block information they store.

1.After synching, DataNode continually poll NameNode for any create , move or delete blocks operations.

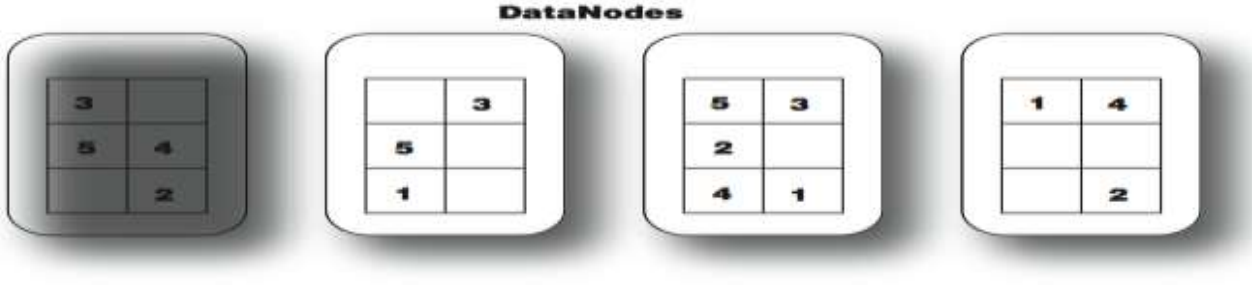


Figure 2.1 NameNode/DataNode interaction in HDFS. The NameNode keeps track of the file metadata—which files are in the system and how each file is broken down into blocks. The DataNodes provide backup store of the blocks and constantly report to the NameNode to keep the metadata current.

Secondary NameNode

- SNN resides on a machine in cluster.
- This machine like that of NN doesn't have any DN or TT daemons running.
- SNN unlike NN, doesn’t record any real time changes.
- Communicates with NN → take snapshot of HDFS cluster and merges the changes.

Secondary NameNode

- Though NN is a single point of failure, with manual interventions , we can minimize the data loss.
- With manual interventions, we ca configure SNN to NN.

JobTracker

- Oversees and coordinates the parallel processing of data using Map Reduce.
- Submit code → determines execution plan by looking at which files to process (data locality is important)
- If a task fails → automatically relaunch the task possibly on a different node.

JobTracker

- Only one jobTracker for a cluster.
- Typically runs on a server as a master node of the cluster.

Task Tracker

- Like data storage, data computations also follow master/slave architecture.
- As DNs are slaves at storage level , TTs are slaves at computation level.

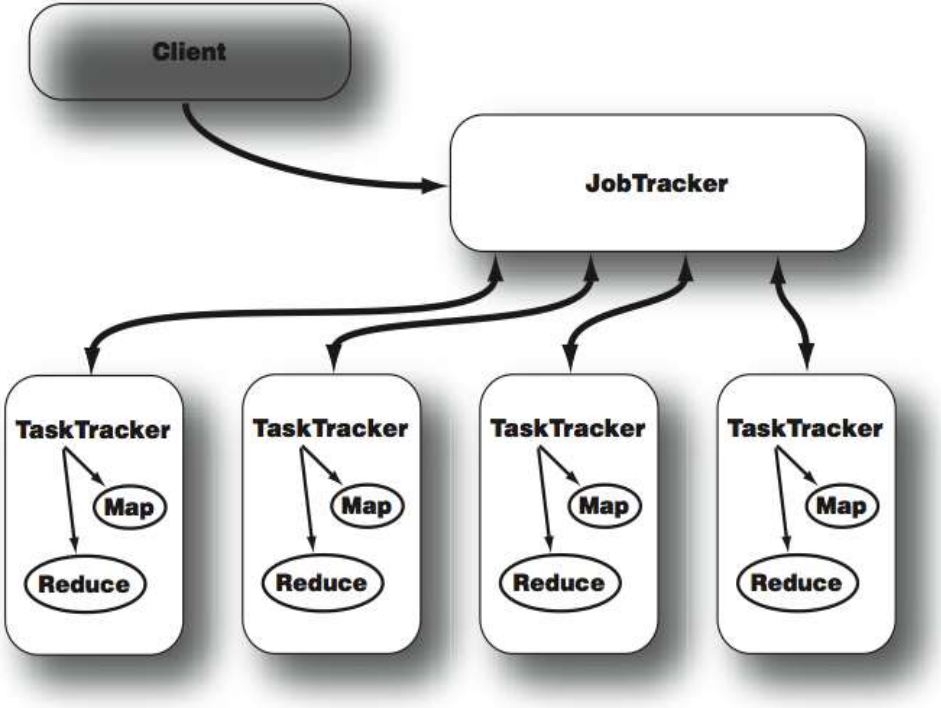
Task Tracker

- JobTracker → over all execution of a job
- TaskTracker is responsible for completion of a task assigned to the node.
- Word of Caution
  - One task tracker per node
  - But task tracker can spawn multiple jvms to handle many mappers and reducers in || .

Task Tracker

- Responsibility of tasktracker to send the heart beat of the task status to jobTracker.
- If jobTracker don't receive heart beat, it assumes that tasktracker got crashed and resubmits the job to another node.

JobTracker and TaskTracker

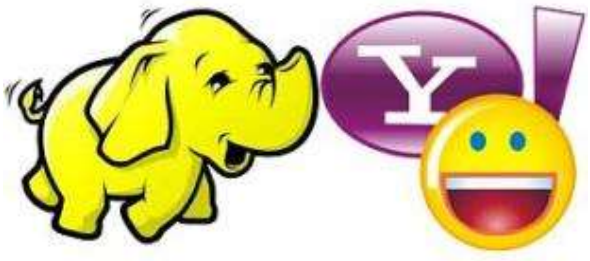


JobTracker and TaskTracker interaction. After a client calls the JobTracker to begin a data processing job, the JobTracker partitions the work and assigns different map and reduce tasks to each TaskTracker in the cluster.

Client

- Not part of cluster
- Used to load data into cluster
- Submit M/R jobs to cluster
- They have hadoop setup but not part of cluster

Yahoo Cluster



- The Yahoo! Search Webmap is a Hadoop application that runs on more than 10,000 core Linux cluster and produces data that is now used in every Yahoo! Web search query.

- On February 19, 2008, Yahoo! Inc. launched what it claimed was the world's largest Hadoop production application

Facebook Cluster

- The Datawarehouse Hadoop cluster at Facebook
- 21 PB of storage in a single HDFS cluster
- 2000 machines
- 12 TB per machine (a few machines have 24 TB each)
- 1200 machines with 8 cores each + 800 machines with 16 cores each
- 32 GB of RAM per machine
- 15 map-reduce tasks per machine
- That's a total of more than 21 PB of configured storage capacity! This is larger than the previously known Yahoo!'s cluster of 14 PB. Here are the cluster statistics from the HDFS cluster at Facebook:

DFShell

The HDFS shell can be invoked by: bin/hadoop dfs <args>

cat
chgrp
chmod
chown
copyFromLocal
copyToLocal
cp
du
dus
expunge
•get
•getmerge
•ls
•lsr
•mkdir
•movefromLocal
•mv
•touchz
•put
•rm
•rmr
•setrep
•stat
•tail
•test
•text

Single Node

Hadoop Cluster Setup

Hadoop Single Node Setup

- Step 1:

  Download hadoop from

  http://hadoop.apache.org/mapreduce/releases.html

- Step 2:

  Untar the hadoop file:

  tar xvfz hadoop-0.20.2.tar.gz

Hadoop Single Node Setup

- Step 3:

Set the path to java compiler by editing

- JAVA_HOME

Parameter in

- hadoop/conf/hadoop---env.sh

Hadoop Single Node Setup

- Step 4:

Create an RSA key to be used by hadoop when ssh'ing to localhost:

```
ssh-keygen -t rsa -P " "
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Hadoop Single Node Setup

- Step 5:

Do the following changes to the configuration files under

- **coresite.xml:**

```
<property>
      <name>fs.default.name</name>
      <value>hdfs://localhost:6000</value>
</property>

<property>
 <name>hadoop.tmp.dir</name>
<value>/Users/ims-a/Documents/hadoop-install/hdfs_location_new/</value>
 <description>A base for other temporary directories.</description>
</property>
```

Hadoop Single Node Setup - mapred-site.xml

- **mapredsite.xml:**

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

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

<configuration>
        <property>
    <name>mapred.tasktracker.map.tasks.maximum</name>
    <value>2</value>
        </property>

        <property>
    <name>mapred.tasktracker.reduce.tasks.maximum</name>
    <value>2</value>
        </property>

        <property>
     <name>mapred.job.tracker</name>
     <value>localhost:6001</value>
        </property>
```

Hadoop Single Node Setup - hdfs-site.xml

- **hdfssite.xml:**

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

Hadoop Single Node Setup

- Step 6:

  Format the hadoop file system. From hadoop directory run the following:

- bin/hadoop namenode -format

Using Hadoop

1)How to start Hadoop?

cd hadoop/bin ./start-all.sh

2)How to stop Hadoop?

cd hadoop/bin ./stop-all.sh

3)How to copy file from local to HDFS?

cd hadoop

bin/hadoop dfs –put local_machine_path hdfs_path

4)How to list files in HDFS?

cd hadoop

bin/hadoop dfs -ls

HADOOP components --more

- Name node
  - High memory requirements
  - Complete file and block metadata in memory
- Secondary NameNode
  - Idle for most of the time
  - Needs memory when creating checkpoint
  - Keeps copy of latest checkpoint of filesystem metadata

HADOOP components --more

- In File System with huge number of files and huge data
  - Not advisable to have namenode and secondary namenode on the same machine
  - Memory constraints
  - SNN purpose itself comes in the case of eventuality of NN

HADOOP components --more

- Masters
  - Secondary NameNodes
- Slaves
  - Data Nodes
  - TaskTrackers

Few startup scripts

- **start-dfs.sh**
  - Starts namenode on the local machine
  - Starts datanode on each machine mentioned in slaves file
  - Starts secondary namenode on each machine mentioned in masters file.

Few startup scripts

- **start-yarn.sh**
  - starts job tracker on the local machine
  - starts task tracker on each machine mentioned in the slaves file.

Few startup scripts

- hadoop-daemon.sh <start/stop> <datanode/namenode/secondarynamenode/job tracker/tasktracker>

What files in cluster in Sync ?

- NameNode and JobTracker on different machines
  - Slaves in NN, JT should be in sync
- Run hdfs scripts from NameNode
- Run MR scripts from JT
- Keeping all the config files in sync across the cluster is a good practice. (exceptions)

Parameters in HADOOP CLUSTER

Environmental Parameters – hadoop-env.sh

MEMORY FOOT PRINT of a SLAVE

- 1Gb for every daemon – default
  - HADOOP_HEAPSIZE
- Task Tracker launch, separate child jvm's for running mappers and reducers
  - Additional memory

nagarjuna@outlook.com

Environmental Parameters – hadoop-env.sh

- How many map tasks for a tasktracker
  - mapred.tasktracker.map.tasks.maximum – 2 default
- How many reduce tasks for a tasktracker
  - mapred.tasktracker.reduce.tasks.maximum – 2 default

Environmental Parameters – hadoop-env.sh

- Memry for child JVM's
  - Mapred.child.java.opts -- default 200 MB

Task Tracker	1MB
Data Node	1000 MB
2- Mappers	2*200 = 400 MB
2- Reducers	2*200 = 400 MB
Total	2800 MB = **2.8 GB**