www.JntukMaterials.com

Unit 2

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

The Google File System

- The Google File System(GFS) is a scalable distributed file system for large distributed data intensive applications.
- It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.
- GFS provides a familiar file system interface, though it does not implement a standard API such as POSIX.
- Files are organized hierarchically in directories and identified by path-names. GFS support the usual operations such as create, delete, open, close, read, and write files.
- GFS has snapshot and record append operations. Snapshot creates a copy of a file or a directory tree at low cost.
- Record append allows multiple clients to append data to the same file concurrently while guaranteeing the atomicity of each individual clients append

Architecture

- A GFS cluster consists of a single **master** and multiple **chunk servers** and is accessed by **multiple clients,** as shown in the following figure .

sacet

www.JntukMaterials.com

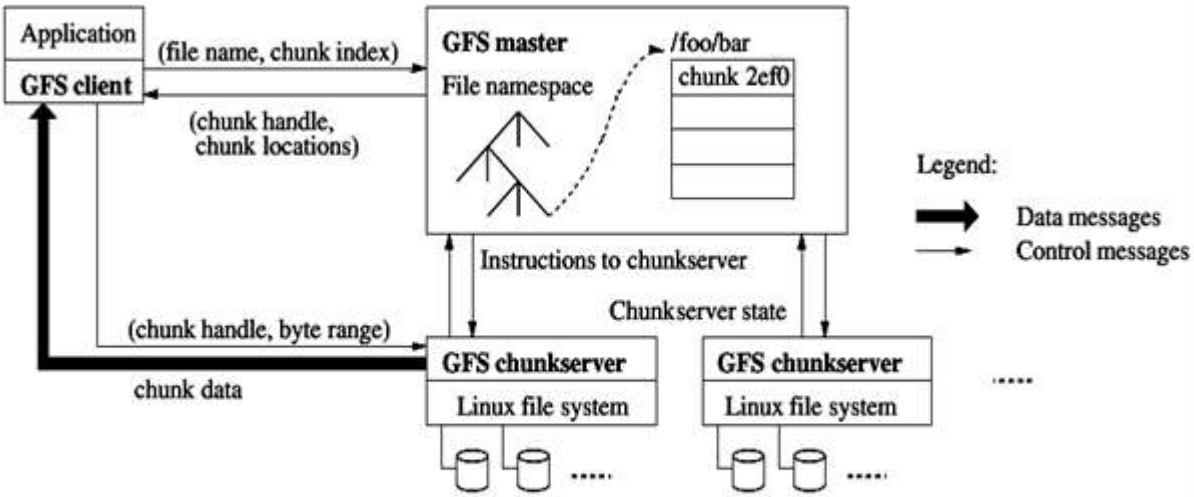

Figure 1: GFS Architecture

- Each of these is typically a commodity Linux machine running a user-level server process.
- Files are divided into fixed-size chunks. Each chunk is identified by a fixed and globally unique **64-bit chunk** handle assigned by the master at the time of chunk creation.
- **Chunk servers** store chunks on local disks as Linux files
- For reliability, each chunk is replicated on multiple chunk servers. By default, there will three replicas and this value can be changed by user
- The **master** maintains all file system metadata. This includes the namespace, access control information, the mapping from files to chunks, and the current locations of chunks.
- It also controls system-wide activities such as chunk lease management, garbage collection of orphaned chunks, and chunk migration between chunk servers.
- The master periodically communicates with each chunk server in **Heart Beat** messages to give it instructions and collect its state.
- GFS **client c**ode linked into each application implements the file system API and communicates with the master and chunk servers to read or write data on behalf of the application.
- **Clients** interact with the master for metadata operations, but all data-bearing communication goes directly to the chunk servers.

Chunk Size

- A large chunk size offers several important **advantages**
- First, it reduces clients' need to interact with the master because reads and writes on the

www.JntukMaterials.com

same chunk requires only one initial request to the master for chunk location information.

- Second, it can reduce network overhead by keeping a persistent TCP connection to the chunkserver over an extended period of time.
- Third, it reduces the size of the metadata stored on the master.

Disadvantages

1. Large block size can lead to **internal fragmentation.**

2. Even with lazy space allocation, a small file consists of a small number of chunks, perhaps just one. The chunk servers storing those chunks may become hot spots if many clients are accessing the same file. In practice, hot spots have not been a major issue because the applications mostly read large multi-chunk files sequentially. To mitigate it, replication and allowance to read from other clients can be done.

sacet

www.JntukMaterials.com

Big Data

- **Big data** is a term for data sets that are so large or complex that traditional data processing application software is inadequate or insufficient to deal with them.
- Big data can be described by the following characteristics:

Volume

- The quantity of generated and stored data. The size of the data determines the value and potential insight- and whether it can actually be considered big data or not.

Variety

- The type and nature of the data. This helps people who analyse it to effectively use the resulting insight. This includes three types:
    - Structured:
        - Structured data refers to information with a high degree of organization, such that inclusion in a relational database is easily and readily searchable by simple, straightforward search engine algorithms or other search operations;
    - Unstructured
        - Unstructured data is essentially the opposite. **Unstructured data** refers to information that either does not have a pre-defined data model or is not organized in a pre-defined manner. Unstructured information is typically text-heavy, but may contain data such as dates, numbers, and facts as well.
    - Semi structured
        - Semi-structured data is information that doesn't reside in a relational database but that does have some organizational properties that make it easier to analyze. With some process we can store them in relation database
        - Examples of semi-structured : CSV but XML and JSON documents are semi structured documents, NoSQL databases are considered as semi structured.

Velocity

- In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.

Variability

- Inconsistency of the data set .

Veracity

- The quality of captured data can vary greatly, affecting accurate analysis.

www.JntukMaterials.com

sacet

www.JntukMaterials.com

Applications of Big Data:

- Big data analysis played a large role in Barack Obama's successful 2012 re-election campaign.
- Big data analysis was tried out for the BJP to win the Indian General Election 2014
- Big data and the IoT work in conjunction. Data extracted from IoT devices provides a mapping of device inter-connectivity.
- eBay.com uses two data warehouses at 7.5 petabytes and 40PB as well as a 40PB Hadoop cluster for search, consumer recommendations, and merchandising.
- Amazon.com handles millions of back-end operations every day
- Facebook handles 50 billion photos from its user base(1,930,025,943 Facebook active users for more info goto http://www.internetlivestats.com/)
- Big data has its application in fields like Health Care, Retail banking, Marketing etc.

Implementations of Big Data:

Building blocks of Hadoop:

- A fully configured cluster, "running Hadoop" means running a set of daemons, or resident programs, on the different servers in your network.
- These daemons have specific roles; some exist only on one server, some exist across multiple servers.
- The daemons include
  - NameNode
  - DataNode
  - Secondary NameNode
  - JobTracker
  - TaskTracker

NameNode

- Hadoop employs a master/slave architecture for both distributed storage and distributed computation.
- The distributed storage system is called the Hadoop *Distributed File System*, or HDFS.
- The NameNode is the master of HDFS that directs the slave DataNode daemons to perform the low-level I/O tasks.
- The NameNode is the bookkeeper of HDFS; it keeps track of how your files are broken down into file blocks, which nodes store those blocks, and the overall health of the distributed file system.
- The server hosting the NameNode typically doesn't store any user data or perform any computations for a MapReduce program.
- The negative aspect of NameNode is that if the NameNode fail then the entire Hadoop cluster will fail.

www.JntukMaterials.com

DataNode

- Each slave machine in HDFS cluster will host a DataNode daemon to perform the reading and writing HDFS blocks to actual files on the local file system.
- When we want to read or write a HDFS file, the file is broken into blocks and the NameNode will tell your client which DataNode each block resides in. Your client communicates directly with the DataNode daemons to process the local files corresponding to the blocks.
- A DataNode may communicate with other DataNode to replicate its data blocks for redundancy
- Figure 2.1 illustrates the roles of the NameNode and DataNodes.

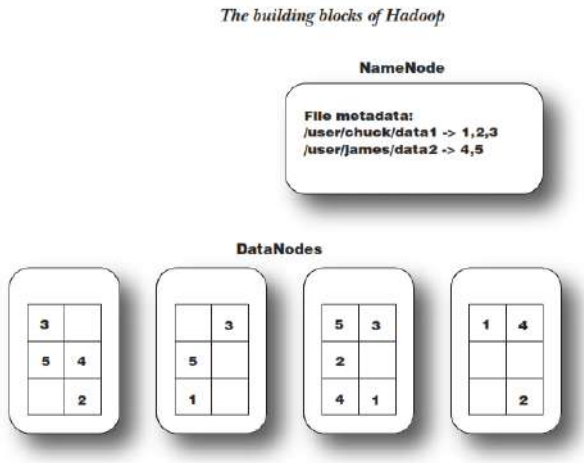

Figure 2.1 NameNode/DataNode interaction in HDFS. The NameNode keeps track of the file metadata—which files are in the system and how each file is broken down into blocks. The DataNodes provide backup store of the blocks and constantly report to the NameNode to keep the metadata current.

- In this figure, we show two data files, one at /user/chuck/data1 and another at /user/james/data2. The data1 file takes up three blocks, which we denote 1, 2, and 3, and the data2 file consists of blocks 4 and 5. The content of the files are distributed among the DataNodes.
- In the above figure each block each has three replicas to ensure that if any one DataNode crashes or becomes inaccessible over the network, we'll still be able to read the files.
- DataNodes are constantly reporting to the NameNode.
- The DataNodes continually communicate with the NameNode to provide information regarding local changes as well as receive instructions to create, move, or delete blocks from the local disk.

Secondary Namenode

- The Secondary NameNode (SNN) is an assistant daemon for monitoring the state of the cluster HDFS.
- Each cluster has one SNN.
- The SNN communicates with the NameNode to take snapshots of the HDFS metadata at intervals defined by the cluster configuration
- The NameNode is a single point of failure for a Hadoop cluster, and the SNN snapshots help minimize the downtime and loss of data
- A NameNode failure requires human involvement to reconfigure the cluster to use the SNN as the primary NameNode

JobTracker

www.JntukMaterials.com

- The JobTracker daemon is the link between your application and Hadoop.
- Once we submit our code to the cluster, the JobTracker determines the execution plan by determining which files to process, assigns nodes to different tasks, and monitors all tasks as they're running.
- If a task fail, the JobTracker will automatically re-launch the task, possibly on a different node, up to a predefined limit of retries.
- There is only one JobTracker daemon per Hadoop cluster. It's typically run on a server as a master node of the cluster.

TaskTracker

- Just like the storage daemons, the computing daemons also follow a master/slave architecture: the JobTracker is the master overseeing the overall execution of a MapReduce job and the TaskTrackers manage the execution of individual tasks on each slave node.
- Figure 2.2 illustrates this interaction

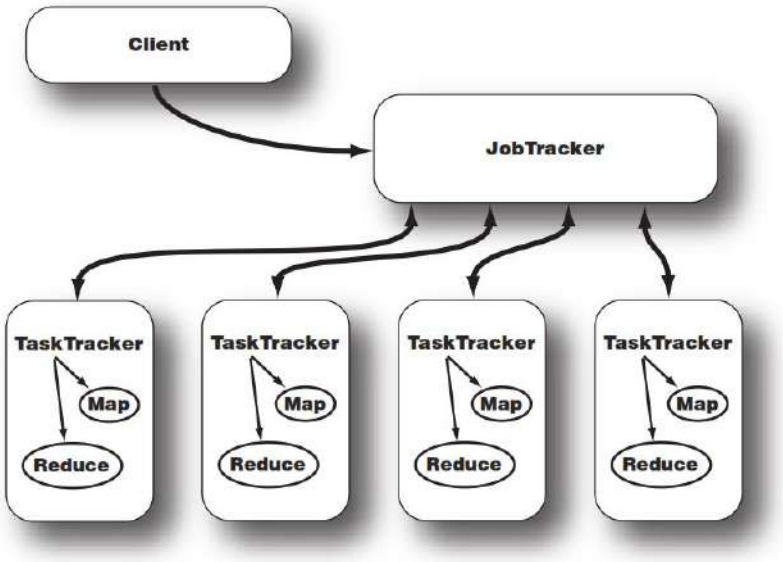

Figure 2.2 JobTracker and TaskTracker interaction. After a client calls the JobTracker to begin a data processing job, the JobTracker partitions the work and assigns different map and reduce tasks to each TaskTracker in the cluster.

- One responsibility of the TaskTracker is to constantly communicate with the JobTracker. If the JobTracker fails to receive a **heartbeat** from a TaskTracker within a specified amount of time, it will assume the TaskTracker has crashed and will resubmit the corresponding tasks to other nodes in the cluster.
- The following figure depict the topology of one typical Hadoop cluster in figure 2.3.

www.JntukMaterials.com

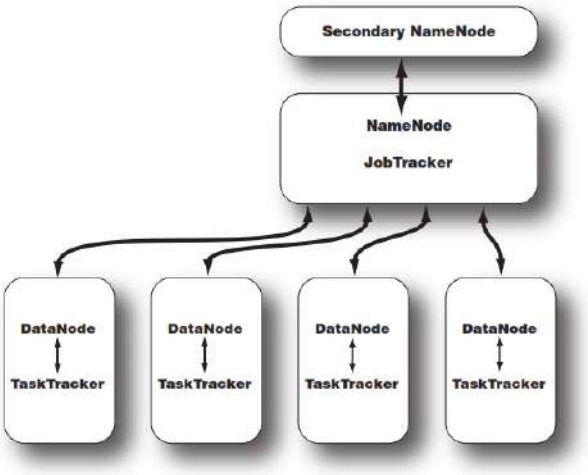

Figure 2.3 Topology of a typical Hadoop cluster. It's a master/slave architecture in which the NameNode and JobTracker are masters and the DataNodes and TaskTrackers are slaves.

- This topology features a master node running the NameNode and JobTracker daemons and a standalone node with the SNN in case the master node fails.
- The slave machines each host a DataNode and TaskTracker, for running tasks on the same node where their data is stored.
-

Introducing and Configuring Hadoop cluster:

- The majority of Hadoop settings are contained in XML configuration files.
- In order to create a Hadoop cluster we need to configure several xml files.
- All the configuration files are stored in **conf** directory of **HADOOP_HOME**.

[hadoop-user@master]$ cd $HADOOP_HOME

[hadoop-user@master]$ ls -l conf/ total 100

-rw-rw-r-- 1 hadoop-user hadoop 2065 Dec 1 10:07 capacity-scheduler.xml

-rw-rw-r-- 1 hadoop-user hadoop 535 Dec 1 10:07 configuration.xsl

-rw-rw-r-- 1 hadoop-user hadoop 49456 Dec 1 10:07 hadoop-default.xml

-rwxrwxr-x 1 hadoop-user hadoop 2314 Jan 8 17:01 hadoop-env.sh

-rw-rw-r-- 1 hadoop-user hadoop 2234 Jan 2 15:29 hadoop-site.xml

-rw-rw-r-- 1 hadoop-user hadoop 2815 Dec 1 10:07 log4j.properties

-rw-rw-r-- 1 hadoop-user hadoop 28 Jan 2 15:29 masters

-rw-rw-r-- 1 hadoop-user hadoop 84 Jan 2 15:29 slaves

-rw-rw-r-- 1 hadoop-user hadoop 401 Dec 1 10:07 sslinfo.xml.example

- In hadoop-env.sh define the JAVA_HOME environment variable to point to the Java installation directory

```
export JAVA_HOME=/usr/share/jdk
```

www.JntukMaterials.com

- Before version 0.20, these XML files are hadoop-default.xml and hadoop-site.xml.
- The hadoop-default.xml contains the default Hadoop settings to be used unless they are explicitly overridden in hadoop-site.xml.
- In version 0.20 the **hadoop-site.xml** file has been separated out into three XML files: **core-site.xml, hdfs-site.xml**, and **mapred-site.xml**.
- A Hadoop cluster can be configured in one of the following 3 modes by modifying the above XML files.
    - Local (standalone) mode
    - Pseudo-distributed mode
    - Fully Distributed mode

Local (standalone) mode :

- The standalone mode is the default mode for Hadoop.
- When we first uncompressed the Hadoop source package, it does not consider our hardware setup. Hadoop chooses to be conservative and assumes a minimal configuration.
- All three XML files (or hadoop- site.xml before version 0.20) are empty under this default mode:

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

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

<configuration>

</configuration>
```

- Its primary use is for developing and debugging the application logic of a MapReduce pro- gram without the additional complexity of interacting with the daemons.

Pseudo-distributed mode

- The pseudo-distributed mode is running Hadoop in a "cluster of one" with all daemons running on a single machine.
- This mode allowing us to examine memory usage, HDFS input/output issues, and other daemon interactions.
- Listing 2.1 provides simple XML files to configure a single server in this mode.

Listing 2.1 Example of the three configuration files for pseudo-distributed mode

core-site.xml

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

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

<configuration>

<property>
 <name>fs.default.name</name>
 <value>hdfs://localhost:9000</value>
 <description>The name of the default file system. A URI whose
 scheme and authority determine the FileSystem implementation.
 </description>
</property>

</configuration>
```

www.JntukMaterials.com

mapred-site.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.job.tracker</name>
 <value>localhost:9001</value>
 <description>The host and port that the MapReduce job tracker runs
 at.</description>
</property>

</configuration>
```

hdfs-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
 <name>dfs.replication</name>
 <value>1</value>
 <description>The actual number of replications can be specified when the file is
 created.</description>
</property>

</configuration>
```

- In core-site.xml and mapred-site.xml we specify the hostname and port of the NameNode and the JobTracker, respectively.
- In hdfs-site.xml we specify the default replication factor for HDFS
- We must also specify the location of the Secondary NameNode in the masters file and the slave nodes in the slaves file:

```
[hadoop-user@master]$ cat masters
localhost
[hadoop-user@master]$ cat slaves
 localhost
```

- While all the daemons are running on the same machine, they still communicate with each other using the same SSH protocol as if they were distributed over a cluster.
- We need to check the machine allows you to `ssh` back to itself.

```
[hadoop-user@master]$ ssh localhost
```

- If the above command results an error then set up `ssh` by using the following commands

```
[hadoop-user@master]$ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
[hadoop-user@master]$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

- Next ,we need to format the namenode by using the following command

```
[hadoop-user@master]$ bin/hadoop namenode -format
```

- To launch the daemons by use of the `start-all.sh` script. The Java `jps` command will list all daemons to verify the setup was successful.

```
[hadoop-user@master]$ bin/start-all.sh
[hadoop-user@master]$ jps
```

www.JntukMaterials.com

```
26893 Jps
26832 TaskTracker
26620 SecondaryNameNode
26333 NameNode
26484 DataNode
26703 JobTracker
```

- We can shut down all the daemons using the command

```
[hadoop-user@master]$ bin/stop-all.sh
```

Fully Distributed mode

- An actual Hadoop cluster runs in the third mode, the fully distributed mode that emphasizing the benefits of distributed storage and distributed computation
- In the discussion below we'll use the following server names:
  - *master*—The master node of the cluster and host of the NameNode and Job- Tracker daemons
  - *backup*—the server that hosts the Secondary NameNode daemon
  - *hadoop1, hadoop2, hadoop3,*—the slave boxes of the cluster running both DataNode and TaskTracker daemons
- Listing 2.2 is a modified version of the pseudo-distributed configuration files (listing 2.1) that can be used as a skeleton for our cluster's setup.

Listing 2.2 Example configuration files for fully distributed mode

core-site.xml

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

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

<configuration>

<property>
 <name>fs.default.name</name>
 <value>hdfs://master:9000</value>
 <description>The name of the default file system. A URI whose
 scheme and authority determine the FileSystem implementation.
 </description>
</property>

</configuration>
```

(1)

mapred-site.xml

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

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

<configuration>
```

www.JntukMaterials.com

```
<property>
 <name>mapred.job.tracker</name>                                    ❷
 <value>master:9001</value>
 <description>The host and port that the MapReduce job tracker runs
 at.</description>
</property>

</configuration>
```

hdfs-site.xml

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

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

<configuration>

<property>
 <name>dfs.replication</name>                                       ❸
 <value>3</value>
 <description>The actual number of replications can be specified when the
 file is created.</description>
</property>

</configuration>
```

The key differences are

- We explicitly stated the hostname for location of the NameNode ❶ and JobTracker ❷ daemons.
- We increased the HDFS replication factor to take advantage of distributed storage ❸. Recall that data is replicated across HDFS to increase availability and reliability.

- We also need to update the masters and slaves files to reflect the locations of the other daemons.

```
[hadoop-user@master]$ cat masters

backup
[hadoop-user@master]$ cat slaves
hadoop1
hadoop2
hadoop3
...
```

- We to format HDFS to prepare it for storage:

```
[hadoop-user@master]$ bin/hadoop namenode - format
```

- Now we can start the Hadoop daemons:

```
[hadoop-user@master]$ bin/start-all.sh
```

- To Verify the nodes are running their assigned jobs.

```
[hadoop-user@master]$ jps

30879 JobTracker

30717 NameNode

30965 Jps

[hadoop-user@backup]$ jps
2099 Jps
```

www.JntukMaterials.com

```
1679 SecondaryNameNode
[hadoop-user@hadoop1]$ jps
7101 TaskTracker
7617 Jps
6988 DataNode
```

sacet