Hadoop

What is Hadoop?

- A software framework that supports data-intensive distributed applications.
- It enables applications to work with **thousands of nodes** and **petabytes of data**.
- Hadoop was inspired by Google's MapReduce and Google File System (GFS).
- Hadoop is a top-level Apache project being built and used by a global community of contributors, using the Java programming language.
- Yahoo! has been the largest contributor to the project, and uses Hadoop extensively across its businesses.

Hadoop

Who uses Hadoop?

YAHOO! facebook twitter

LinkedIn ebay IBM amazon

AOL Adobe Bai du 图标

last·fm hulu

Hadoop

Who uses Hadoop?

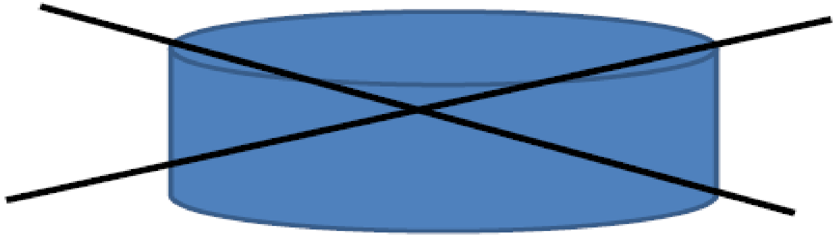
- Yahoo!
  - More than 100,000 CPUs in >36,000 computers.

- Facebook
  - Used in reporting/analytics and machine learning and also as storage engine for logs.
  - A 1100-machine cluster with 8800 cores and about 12 PB raw storage.
  - A 300-machine cluster with 2400 cores and about 3 PB raw storage.
  - Each (commodity) node has 8 cores and 12 TB of storage.

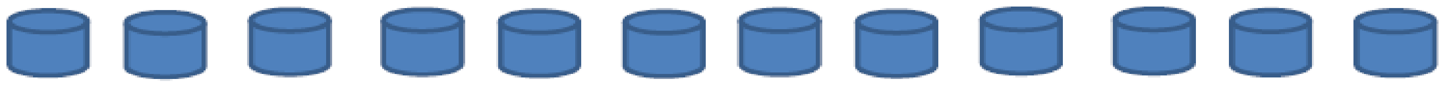
Hadoop

Very Large Storage Requirements

- Facebook has Hadoop clusters with 15 PB of raw storage (15,000,000 GB).
- No single storage can handle this amount of data.



- We need a large set of nodes each storing part of the data.



Introduction

- Distributed programming framework.
- Hadoop is an open source framework for writing and running distributed applications that process large amounts of data.
- Key points of hadoop are
- ***Accessible***
  - *Hadoop runs on large clusters of commodity machines or on cloud computing services such as Amazon’s Elastic Compute Cloud (EC2).*
- ***Robust***
  - *Because it is intended to run on commodity hardware, Hadoop is architected with the assumption of frequent hardware malfunctions. It can gracefully handle most such failures.*
- ***Scalable***
  - *Hadoop scales linearly to handle larger data by adding more nodes to the cluster.*
- ***Simple***
  - *Hadoop allows users to quickly write efficient parallel code.*

Introduction

- Hadoop consist of distributed file system, called HDFS.
- The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware.
- HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware.
- HDFS provides high throughput access to application data and is suitable for applications that have large data sets.
  - HDFS was originally built as infrastructure for the Apache Nutch web search engine project.
  - HDFS is part of the Apache Hadoop Core project.

Hadoop Daemons

- *Hadoop consist of five daemons.*
  - ***NameNode***
  - ***DataNode***
  - ***Secondary nameNode***
  - ***Job tracker***
  - ***Task tracker***

- *"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.*

Name Node

- Hadoop employs a master/slave architecture for both distributed storage and distributed computation.
- The distributed storage system is called the *Hadoop 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 function of the NameNode is memory and I/O intensive. As such, the server hosting the NameNode typically doesn't store any user data or perform any computations for a MapReduce program to lower the workload on the machine.
- it's a single point of failure of your Hadoop cluster.
- For any of the other daemons, if their host nodes fail for software or hardware reasons, the Hadoop cluster will likely continue to function smoothly or you can quickly restart it. Not so for the NameNode.

DataNode

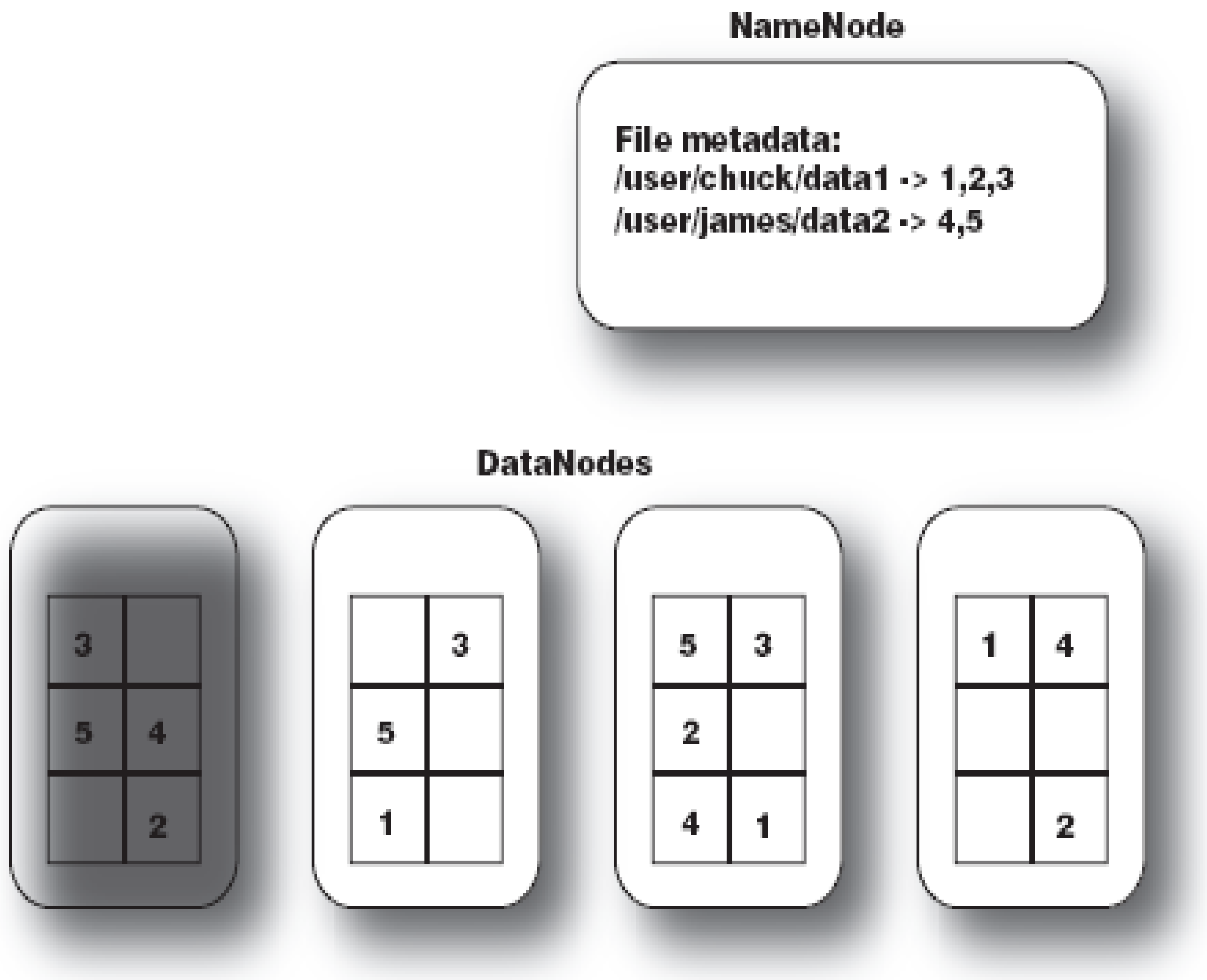
- Each slave machine in cluster host a DataNode daemon to perform work of the distributed file system, reading and writing HDFS blocks to actual files on the local file system.

- 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.

- Job communicates directly with the DataNode daemons to process the local files corresponding to the blocks.

- Furthermore, a DataNode may communicate with other DataNodes to replicate its data blocks for redundancy.

DataNode


NameNode
File metadata:
/user/chuck/data1 -> 1,2,3
/user/james/data2 -> 4,5
DataNodes
3
5 4
2
3
5
1
5 3
2
4 1
1 4
2

DataNode

- DataNodes are constantly reporting to the NameNode.
- Upon initialization, each of the DataNodes informs the NameNode of the blocks it’s currently storing.
- After this mapping is complete, the DataNodes continually poll 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.
- Like the NameNode, each cluster has one SNN.
- No other DataNode or TaskTracker daemons run on the same server.
- The SNN differs from the NameNode, it doesn’t receive or record any real-time changes to HDFS.
- Instead, it communicates with the NameNode to take snapshots of the HDFS metadata at intervals defined by the cluster configuration.
- As mentioned earlier, the NameNode is a single point of failure for a Hadoop cluster, and the SNN snapshots help minimize the downtime and loss of data.
- Nevertheless, a NameNode failure requires human intervention to reconfigure the cluster to use the SNN as the primary NameNode.

JobTracker

- The JobTracker daemon is the liaison (mediator) between your application and Hadoop.
- Once you submit your code to your 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.
- Should 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

Task tracker

- Job Tracker is the master overseeing the overall execution of a Map-Reduce job.

- Task Trackers manage the execution of individual tasks on each slave node.

- Each Task Tracker is responsible for executing the individual tasks that the Job Tracker assigns.

- Although there is a single Task Tracker per slave node, each Task Tracker can spawn multiple JVMs to handle many map or reduce tasks in parallel.

Hadoop Master/Slave Architecture

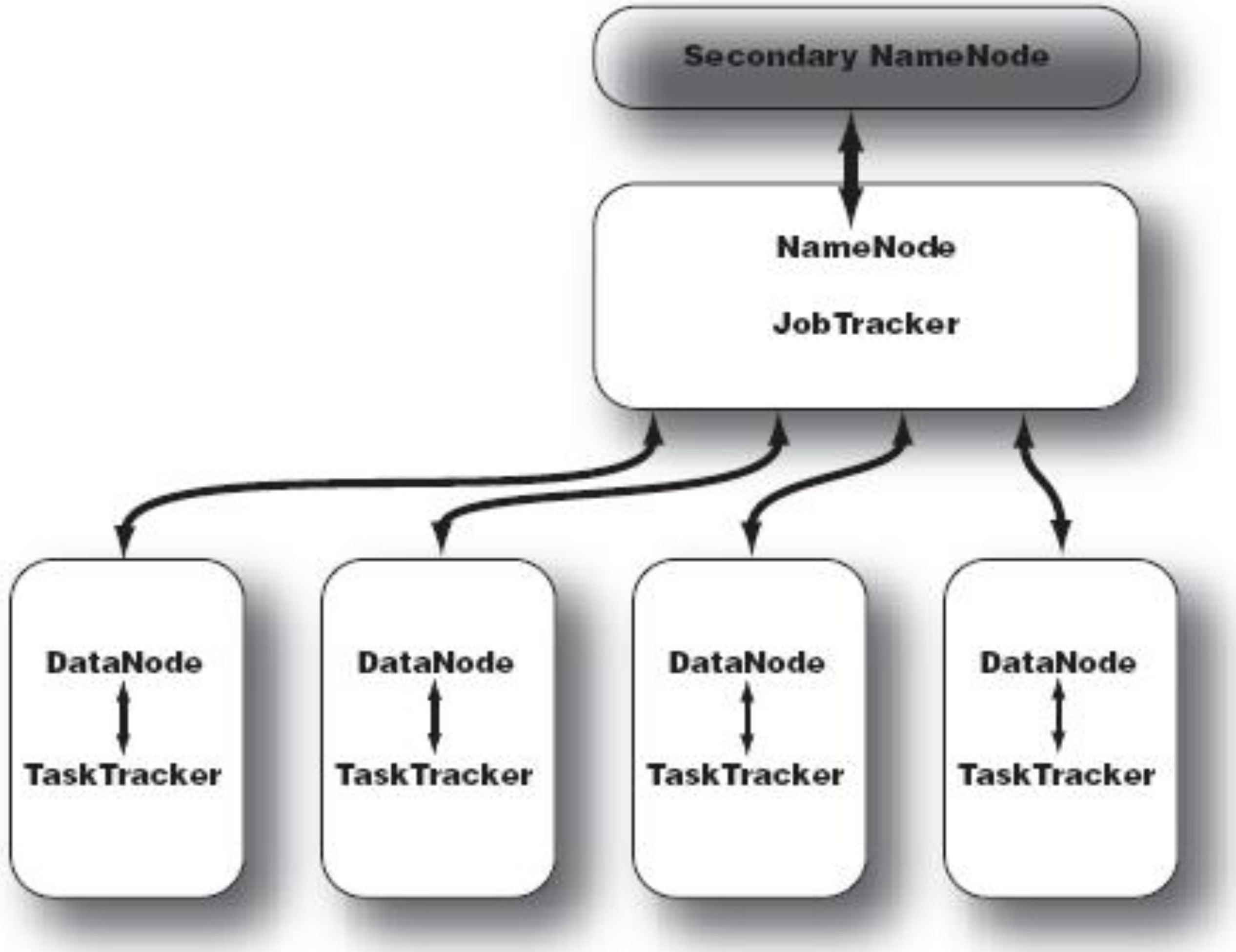


Figure 2.3 Topology of a typical Hadoop cluster. It's a master/slave architecture in which the NameNode and JobTracker are masters and the DataNodes and TaskTrackers are slaves.

Master/slave Architecture

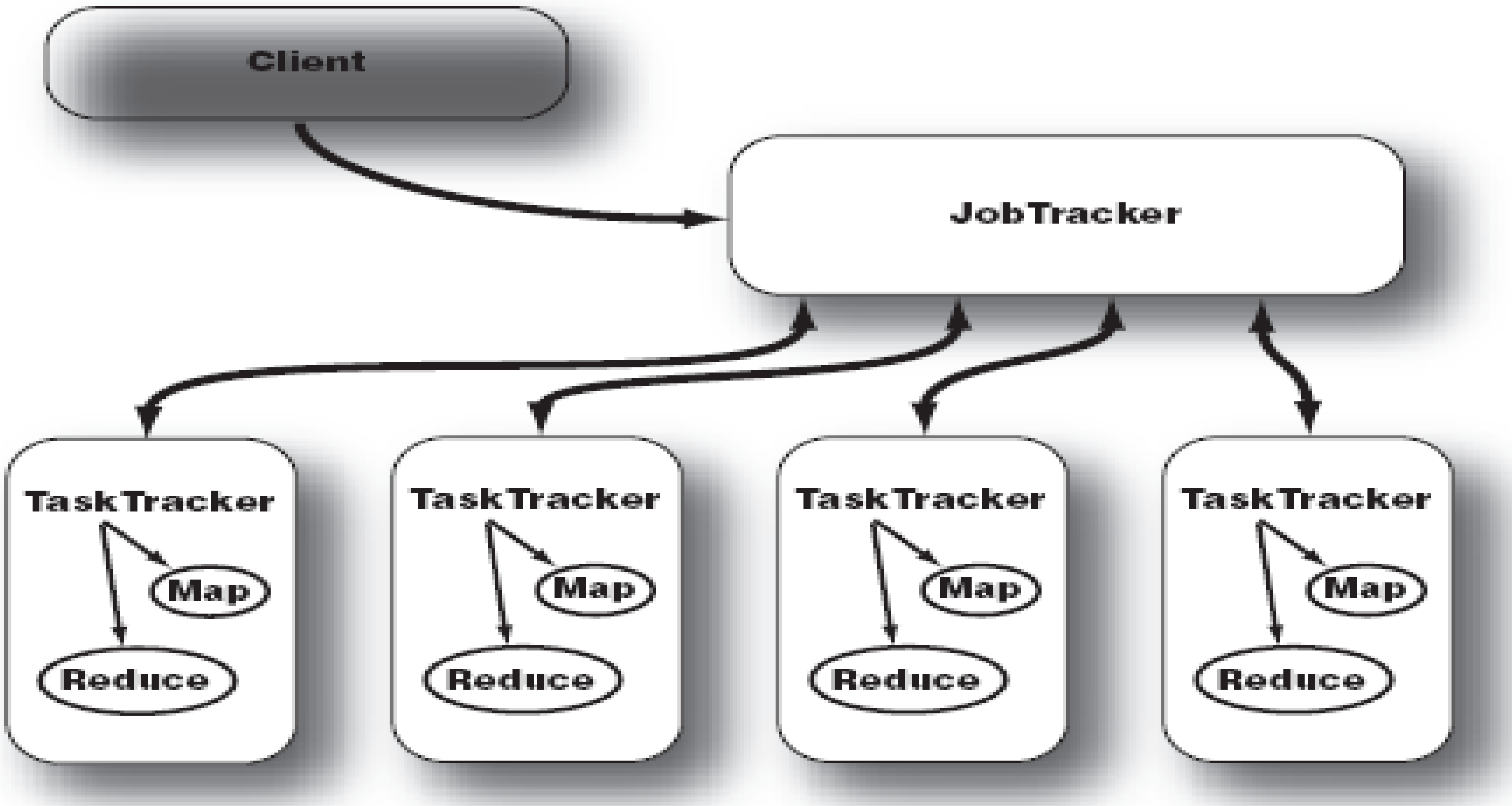


Figure 2.2 JobTracker and TaskTracker interaction. After a client calls the JobTracker to begin a data processing job, the JobTracker partitions the work and assigns different map and reduce tasks to each TaskTracker in the cluster.

Hadoop Configuration Modes

- ***Local (standalone) mode***
- The standalone mode is the default mode for Hadoop.
- Hadoop chooses to be conservative and assumes a minimal configuration. All XML (Configuration) files are empty under this default mode.
- With empty configuration files, Hadoop will run completely on the local machine.
- Because there’s no need to communicate with other nodes, the standalone mode doesn’t use HDFS, nor will it launch any of the Hadoop daemons.
- Its primary use is for developing and debugging the application logic of a Map-Reduce program without the additional complexity of interacting with the daemons.

Hadoop Configuration Modes

- ***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 complements the standalone mode for debugging your code, allowing you to examine memory usage, HDFS input/output issues, and other daemon interactions.
- Need Configuration on XML Files hadoop/conf/.

Hadoop Configuration Modes

- ***Configuration***
- **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>
- **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>

Hadoop Configuration Modes

- ***Fully distributed mode***
- Benefits of distributed storage and distributed computation
- ***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*

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
...
```

Once you have copied these files across all the nodes in your cluster, be sure to format HDFS to prepare it for storage:

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

Now you can start the Hadoop daemons:

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

and verify the nodes are running their assigned jobs.

```
[hadoop-user@master]$ jps
30879 JobTracker
30717 NameNode
```

Working with files in HDFS

- HDFS is a file system designed for large-scale distributed data processing under frameworks such as Map-Reduce.
- Store a big data set of (say) 100 TB as a single file in HDFS.
- Replicate the data for availability and distribute it over multiple machines to enable parallel processing.
- HDFS abstracts these details away and gives you the illusion that you're dealing with only a single file.
- Hadoop Java libraries for handling HDFS files programmatically.

Hadoop Shell commands (***Basic file commands***)

- ***Basic file commands***
- Hadoop file commands take the form of
- **hadoop fs -*cmd <args>***
  - ***hadoop fs –ls (****list the hdfs content****)***
  - ***hadoop fs –mkdir /user/hdfs/test*** *(make directory)*
  - ***hadoop fs –rmr /user/hdfs/test*** *(delete directory/files)*
  - **hadoop fs -cat example.txt | head**
  - ***hadoop fs –copyFromLocal Desktop/test.txt /user/hdfs/***
  - ***hadoop fs –copyToLocal /user/hdfs/test.txt Desktop/***
  - ***Etc…***

Assumptions and Goals

- ***Hardware Failure***
- *Hardware failure is the norm rather than the exception.*
- *An HDFS instance may consist of hundreds or thousands of server machines, each storing part of the file system's data.*
- *The fact that there are a huge number of components and that each component has a non-trivial probability of failure means that some component of HDFS is always non-functional.*
- *Therefore, detection of faults and quick, automatic recovery from them is a core architectural goal of HDFS.*

Assumptions and Goals

- ***Streaming Data Access***
- *Applications that run on HDFS need streaming access to their data sets.*
- *They are not general purpose applications that typically run on general purpose file systems.*
- *HDFS is designed more for batch processing rather than interactive use by users.*
- *The emphasis is on high throughput of data access rather than low latency of data access.*

Assumptions and Goals

- ***Large Data Sets***
- *Applications that run on HDFS have large data sets.*
- *A typical file in HDFS is gigabytes to terabytes in size. Thus, HDFS is tuned to support large files.*
- *It should provide high aggregate data bandwidth and scale to hundreds of nodes in a single cluster.*
- *It should support tens of millions of files in a single instance*

Assumptions and Goals

- ***Simple Coherency Model***
- *HDFS applications need a write-once-read-many access model for files.*
- *A file once created, written, and closed need not be changed.*
- *This assumption simplifies data coherency issues and enables high throughput data access.*
- *A Map/Reduce application or a web crawler application fits perfectly with this model.*
- *There is a plan to support appending-writes to files in the future.*

Assumptions and Goals

- ***“Moving Computation is Cheaper than Moving Data”***
- *A computation requested by an application is much more efficient if it is executed near the data it operates on.*
- *This is especially true when the size of the data set is huge. This minimizes network congestion and increases the overall throughput of the system.*
- *The assumption is that it is often better to migrate the computation closer to where the data is located rather than moving the data to where the application is running.*
- *HDFS provides interfaces for applications to move themselves closer to where the data is located.*

Assumptions and Goals

- ***Portability Across Heterogeneous Hardware and Software Platforms***
- *HDFS has been designed to be easily portable from one platform to another.*
- *This facilitates widespread adoption of HDFS as a platform of choice for a large set of applications.*

Assignment

- Interrelation between hadoop and Amazon cloud.

- Q&A/Feedback?