

Big Data Analytics

Lecture 2



Yilu Zhou, PhD
Associate Professor
Gabelli School of Business
Fordham University

* Some slides are adopted from Professor Kunpeng Zhang's Big Data course @UMD

What we'll cover...

- Hadoop history and advantages
- Architecture
 - Hadoop distributed file system (HDFS)
- Hands-on **Preparation for Cloudera environment**
 - Installation of GitHub – Lab 1
 - Basic Linux commands – Lab 2

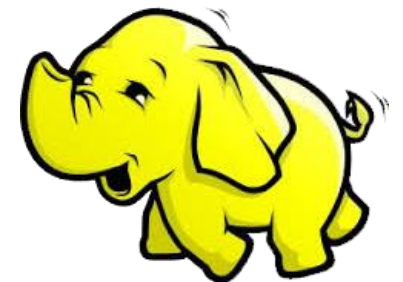
Brief History of Hadoop

Designed to answer the question:

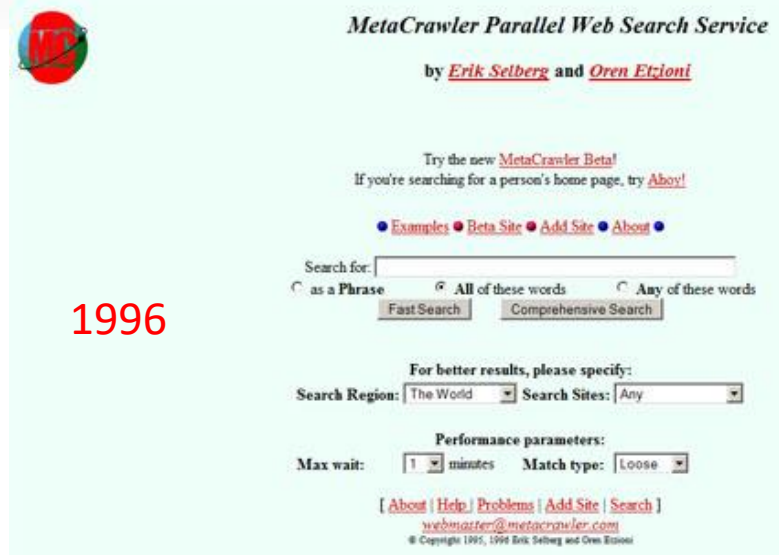
“How to process big data with reasonable cost and time?”

Slides adopted from:

UCI CS 237 Distributed Systems Middleware, Nalini Venkatasubramanian



Search engines in 1990s



1996



1996

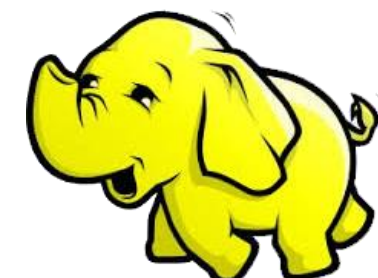
[New Search](#) • [Top News](#) • [Sites by Subject](#) • [Top 5% Sites](#) • [City Guide](#) • [Pictures & Sounds](#)
[PeopleFind](#) • [Point Review](#) • [Road Maps](#) • [Software](#) • [About Lycos](#) • [Club Lycos](#) • [Help](#)
[Add Your Site to Lycos](#)
Copyright © 1996 Lycos™, Inc. All Rights Reserved.
Lycos is a trademark of Carnegie Mellon University.
[Questions & Comments](#)



1996



1997



Slides adopted from:

UCI CS 237 Distributed Systems Middleware, Nalini Venkatasubramanian

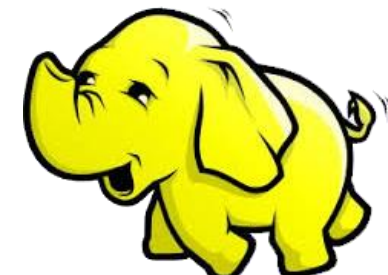
Google search engines



1998



2013



Slides adopted from:

UCI CS 237 Distributed Systems Middleware, Nalini Venkatasubramanian

Early Large Scale Computing

- Historically computation was processor-bound
 - Data volume has been relatively small
 - Complicated computations are performed on that data
- Advances in computer technology has historically centered around improving the power of a single machine

Distributed System: Problems

“You know you have a distributed system when the crash of a computer you’ve never heard of stops you from getting any work done.” –Leslie Lamport

- Distributed systems must be designed with the expectation of failure

Component Recovery

- If a component fails, it should be able to recover without restarting the entire system
- Component failure or recovery during a job must not affect the final output

Hadoop

- Based on work done by **Google** in the early 2000s
 - “The Google File System” in 2003
 - “MapReduce: Simplified Data Processing on Large Clusters” in 2004
- The core idea was to distribute the data as it is initially stored
 - Each node can then perform computation on the data it stores without moving the data for the initial processing

Hadoop's Developers



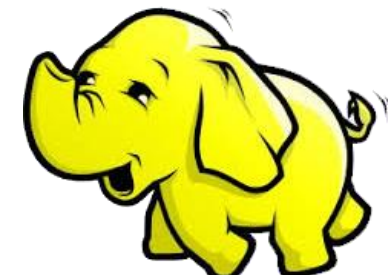
Doug Cutting



2005: Doug Cutting and Michael J. Cafarella developed Hadoop to support distribution for the [Nutch](#) search engine project.

The project was funded by Yahoo.

2006: Yahoo gave the project to Apache Software Foundation.



Slides adopted from:

UCI CS 237 Distributed Systems Middleware, Nalini Venkatasubramanian

Google Origins

2003

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google*



2004

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.



2006

Bigtable: A Distributed Storage System for Structured Data

Fuy Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber
(fuy.jeff.sanjay.wilson.hsieh.chandra.burrows.fikes.gruber}@google.com

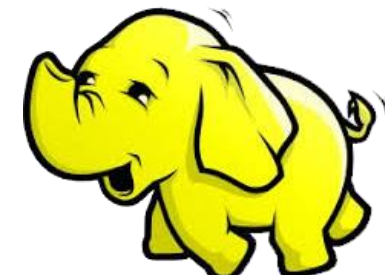
Google, Inc.



Abstract

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large number of servers. Many projects at Google store data in Bigtable, including web indexing, Google Earth, and Google Fused Location Platform. These applications place very different demands on Bigtable, both in terms of data size (from URLs to

achieved scalability and high performance, but Bigtable provides a different interface than such systems. Bigtable does not support a full relational data model; instead, it provides clients with a simple data model that supports dynamic control over data layout and format, and allows clients to reason about the locality properties of data represented in the underlying storage. Data is indexed using row and column names that can be arbitrary strings. Bigtable also treats data as uninterpreted strings.

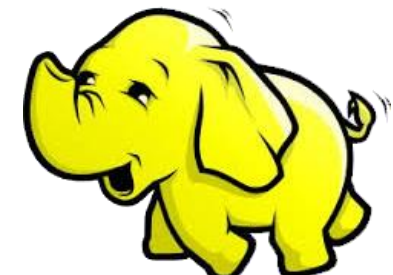


Slides adopted from:

UCI CS 237 Distributed Systems Middleware, Nalini Venkatasubramanian

Some Hadoop Milestones

- **2008 - Hadoop Wins Terabyte Sort Benchmark** (sorted 1 terabyte of data in 209 seconds, compared to previous record of 297 seconds)
- 2009 - Avro and Chukwa became new members of Hadoop Framework family
- 2010 - Hadoop's Hbase, Hive and Pig subprojects completed, adding more computational power to Hadoop framework
- **2011 - ZooKeeper Completed**
- **2013 - Hadoop 1.1.2 and Hadoop 2.0.3 alpha.**
 - Ambari, Cassandra, Mahout have been added



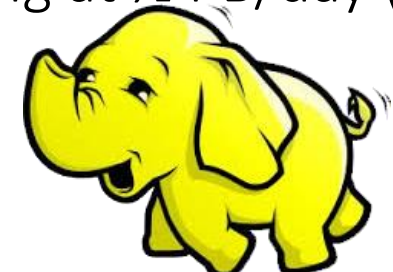
One popular solution: Hadoop



Hadoop Cluster at Yahoo!

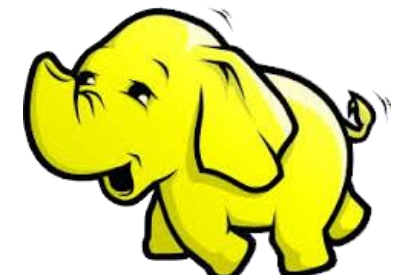
Hadoop in the Wild

- Hadoop is in use at most organizations that handle big data:
 - Yahoo!
 - Facebook
 - Amazon
 - Netflix
 - Etc...
- Some examples of scale:
 - Yahoo!'s Search Webmap runs on 10,000 core Linux cluster and powers Yahoo! Web search
 - FB's Hadoop cluster hosts 100+ PB of data (July, 2012) & growing at ½ PB/day (Nov, 2012)



Three main applications of Hadoop

- Advertisement (Mining user behavior to generate recommendations)
- Searches (group related documents)
- Security (search for uncommon patterns)



Big data storage is challenging

- Data **volumes** are massive
- **Reliability** of storing PBs of data is challenging
- All kinds of **failures**: Disk/Hardware/Network Failures
- Probability of failures simply increase with the number of machines ...

How much data?

- Facebook
 - 500 TB per day
- Yahoo
 - Over 170 PB
- eBay
 - Over 6 PB
- Getting the data to the processors becomes the bottleneck

Hadoop offers

- Redundant, Fault-tolerant data storage
- Parallel computation framework
- Job coordination



Hadoop offers

- Redundant, Fault-tolerant data storage
- Parallel computation framework
- Job coordination



Programmers

*No longer need to
worry about*



**Q: Where file is
located?**

**Q: How to handle
failures & data lost?**

**Q: How to divide
computation?**

**Q: How to program
for scaling?**

Who uses Hadoop?

Social



User Tracking & Engagement



Homeland Security



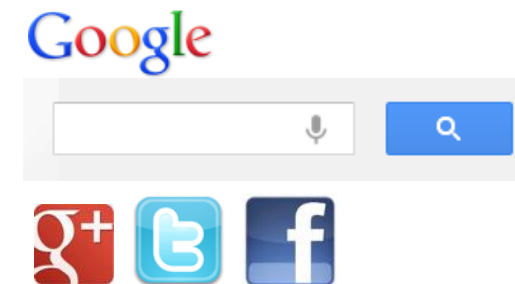
eCommerce






















































































Financial Services



Real Time Search

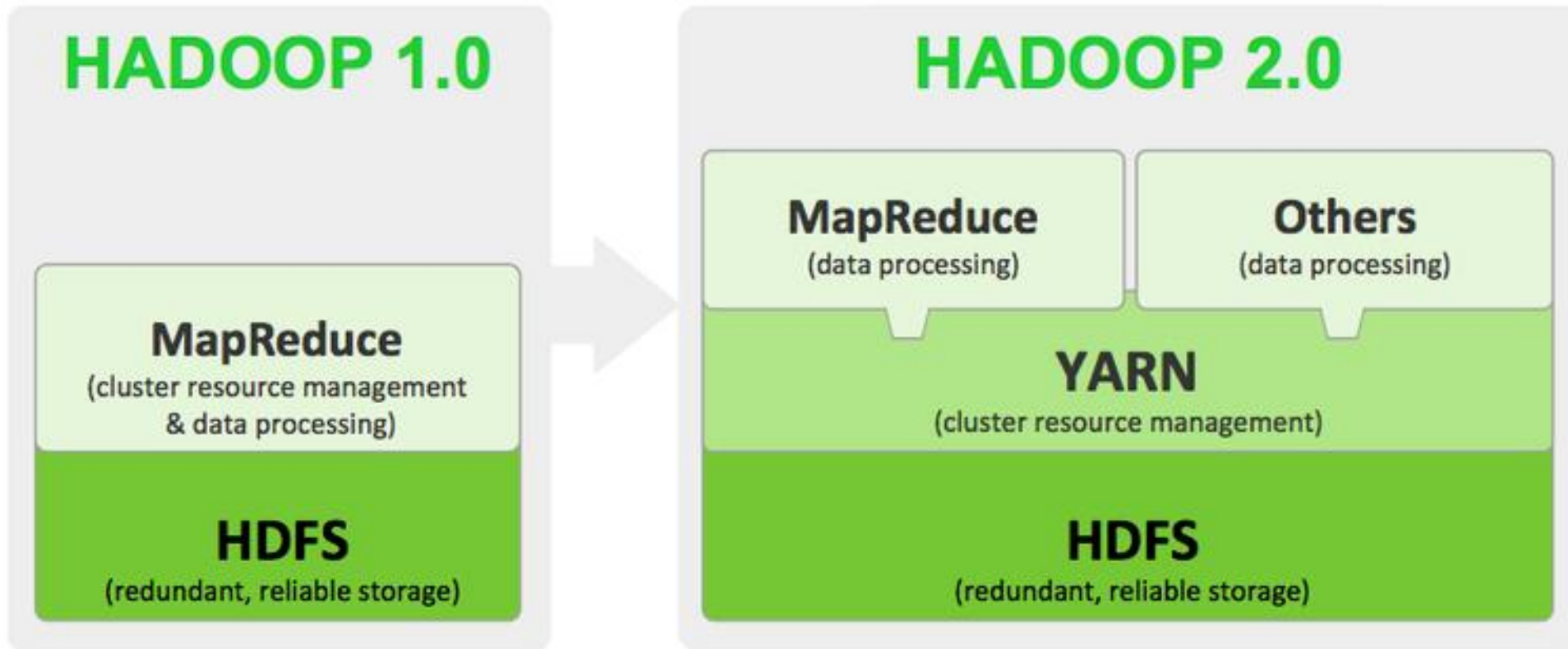


Who uses Hadoop?

2007	2008	2009	2010
  	                      	                         	                              

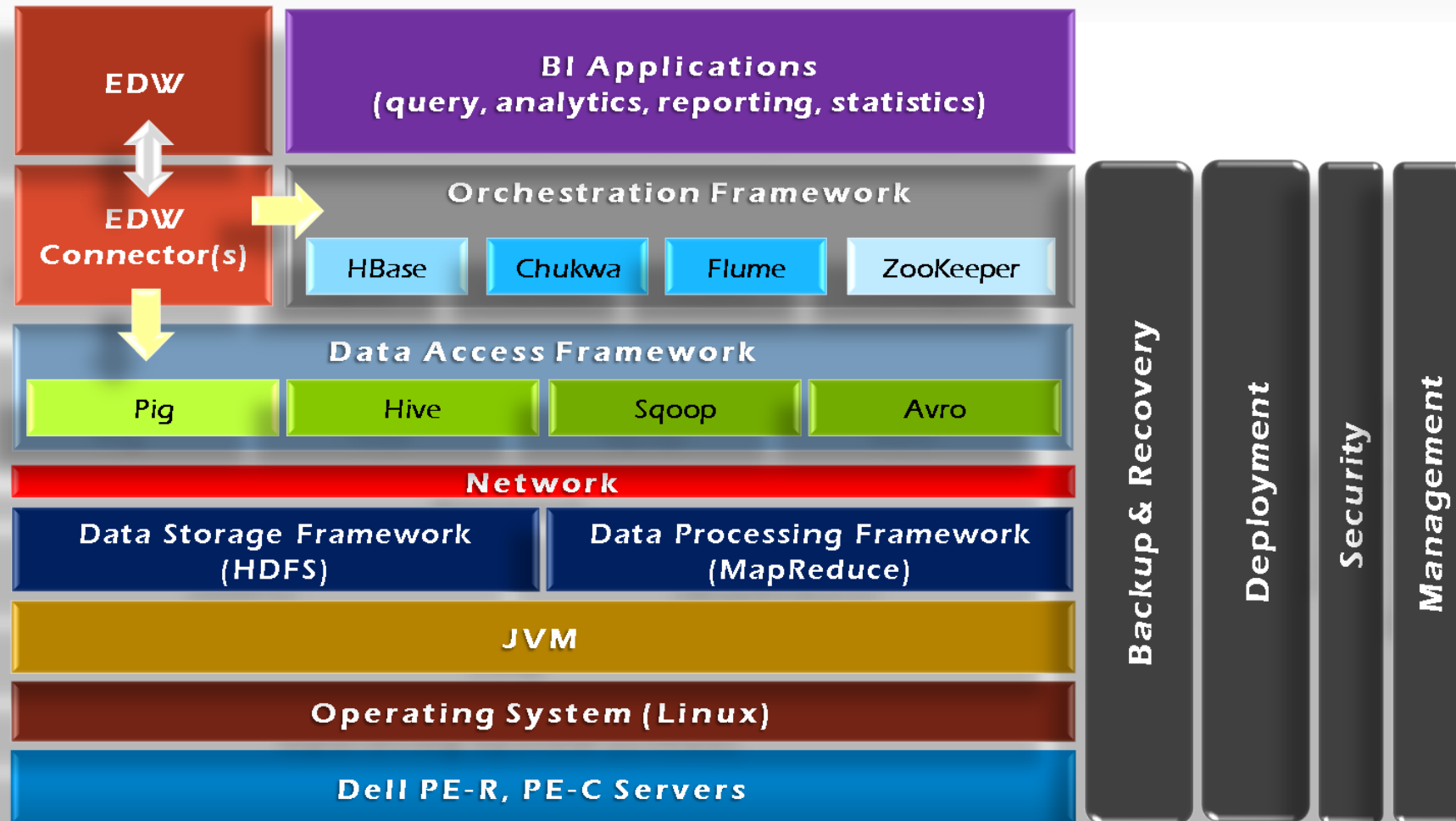
Hadoop big picture

MapReduce: Programming
YARN: Resource Scheduling
Hadoop: Data Management



HDFS + MapReduce is core to get work done

Hadoop Framework Tools



Hadoop goals

- **Scalable**: Petabytes (10^{15} Bytes) of data on thousands on nodes
- **Economical**: Commodity components only
- **Reliable**: fault tolerance

Hadoop Distributed File System

HDFS

Overview

- Responsible for storing data on the cluster
- Data files are split into blocks and distributed across the nodes in the cluster
- Each block is replicated multiple times

HDFS Basic Concepts

- HDFS is a file system written in Java based on the Google's GFS
- Provides redundant storage for massive amounts of data

HDFS

- Master-Slave architecture
- Single NameNode
 - Sometimes a backup: secondary NameNode
- Many (Thousands) DataNodes
- Files are split into fixed sized blocks and stored on data nodes (default: 64MB)
- Data blocks are replicated for fault tolerance and fast access (default: 3)

HDFS Basic Concepts

- HDFS works best with a smaller number of large files
 - Millions as opposed to billions of files
 - Typically 100MB or more per file
- Files in HDFS are write once
- Optimized for streaming reads of large files and not random reads

How are Files Stored

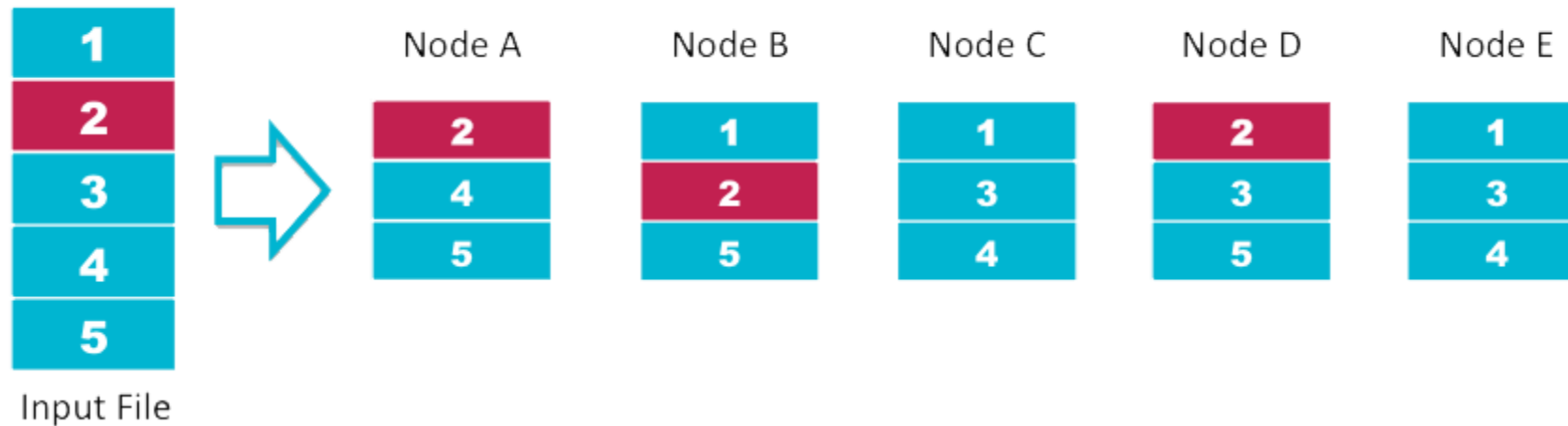
- Files are split into blocks
- Blocks are split across many machines at load time
 - Different blocks from the same file will be stored on different machines
- Blocks are replicated across multiple machines
- The NameNode keeps track of which blocks make up a file and where they are stored

Data Replication

- Default replication is 3-fold

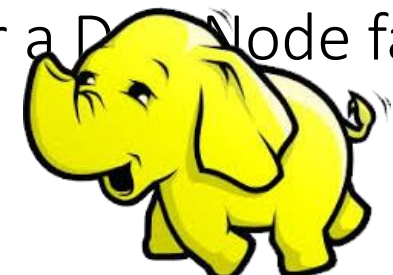
copies not store in the same node

HDFS Data Distribution



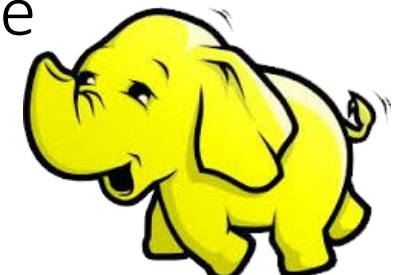
Hadoop's Architecture

- NameNode:
- Stores metadata for the files, like the directory structure of a typical FS.
the location of the node
- The server holding the NameNode instance is quite crucial, as there is only one.
- Transaction log for file deletes/adds, etc. Does not use transactions for whole blocks or file-streams, only metadata.
- Handles creation of more replica blocks when necessary after a DataNode failure

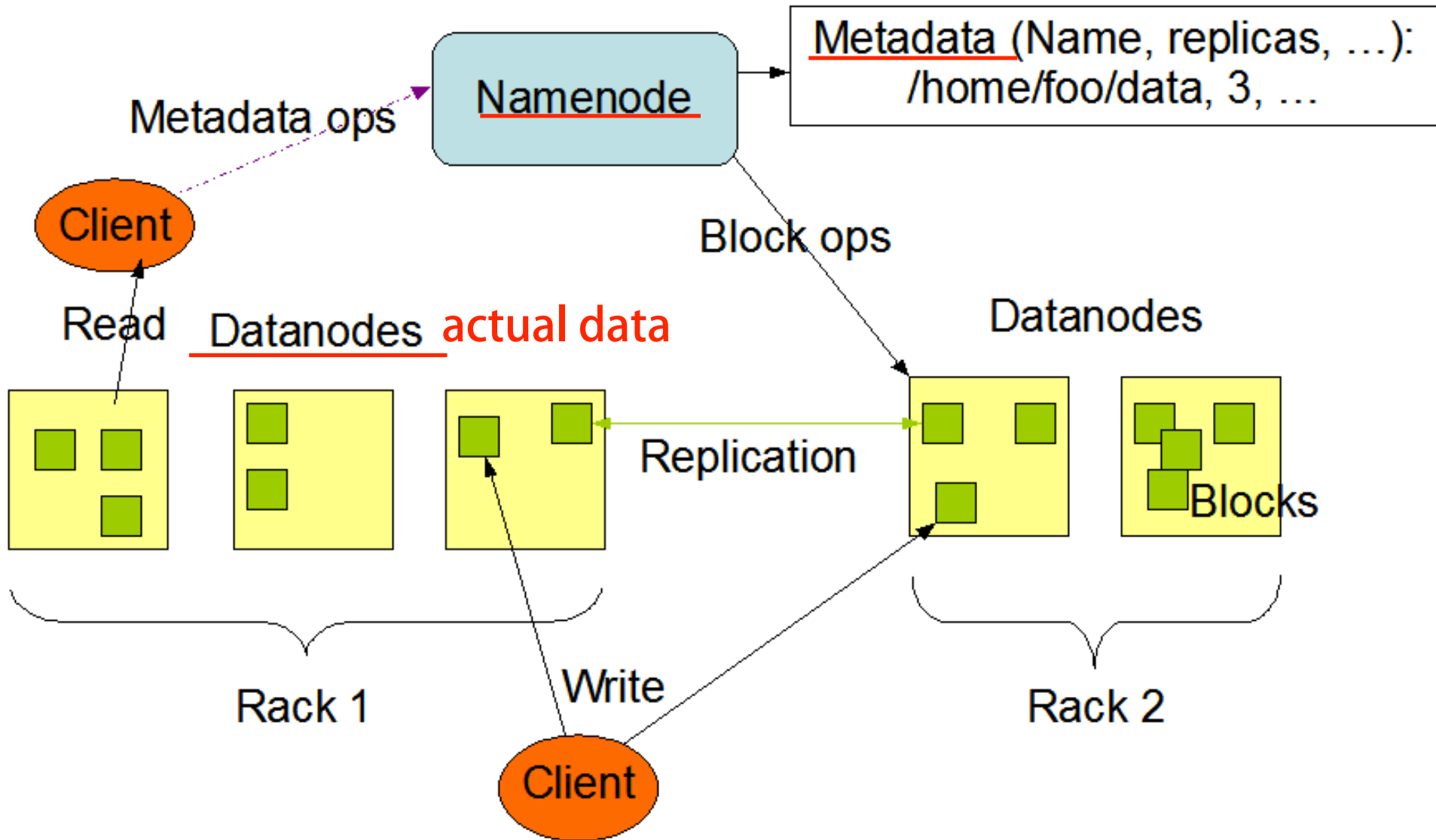


Hadoop's Architecture

- DataNode:
- Stores the actual data in HDFS
- Can run on any underlying filesystem (ext3/4, NTFS, etc)
- Notifies NameNode of what blocks it has
- NameNode replicates blocks 2x in local rack, 1x elsewhere



HDFS Architecture



Data Retrieval

- When a client wants to retrieve data
 - Communicates with the NameNode to determine which blocks make up a file and on which data nodes those blocks are stored
 - Then communicated directly with the data nodes to read the data

MapReduce

Distributing computation across nodes

MapReduce Overview

- A method for distributing computation across multiple nodes
- Each node processes the data that is stored at that node
- Consists of two main phases
 - Map
 - Reduce

MapReduce Features

- Automatic parallelization and distribution
- Fault-Tolerance
- Provides a clean abstraction for programmers to use



The Mapper

- Reads data as key/value pairs
 - The key is often discarded
- Outputs zero or more key/value pairs

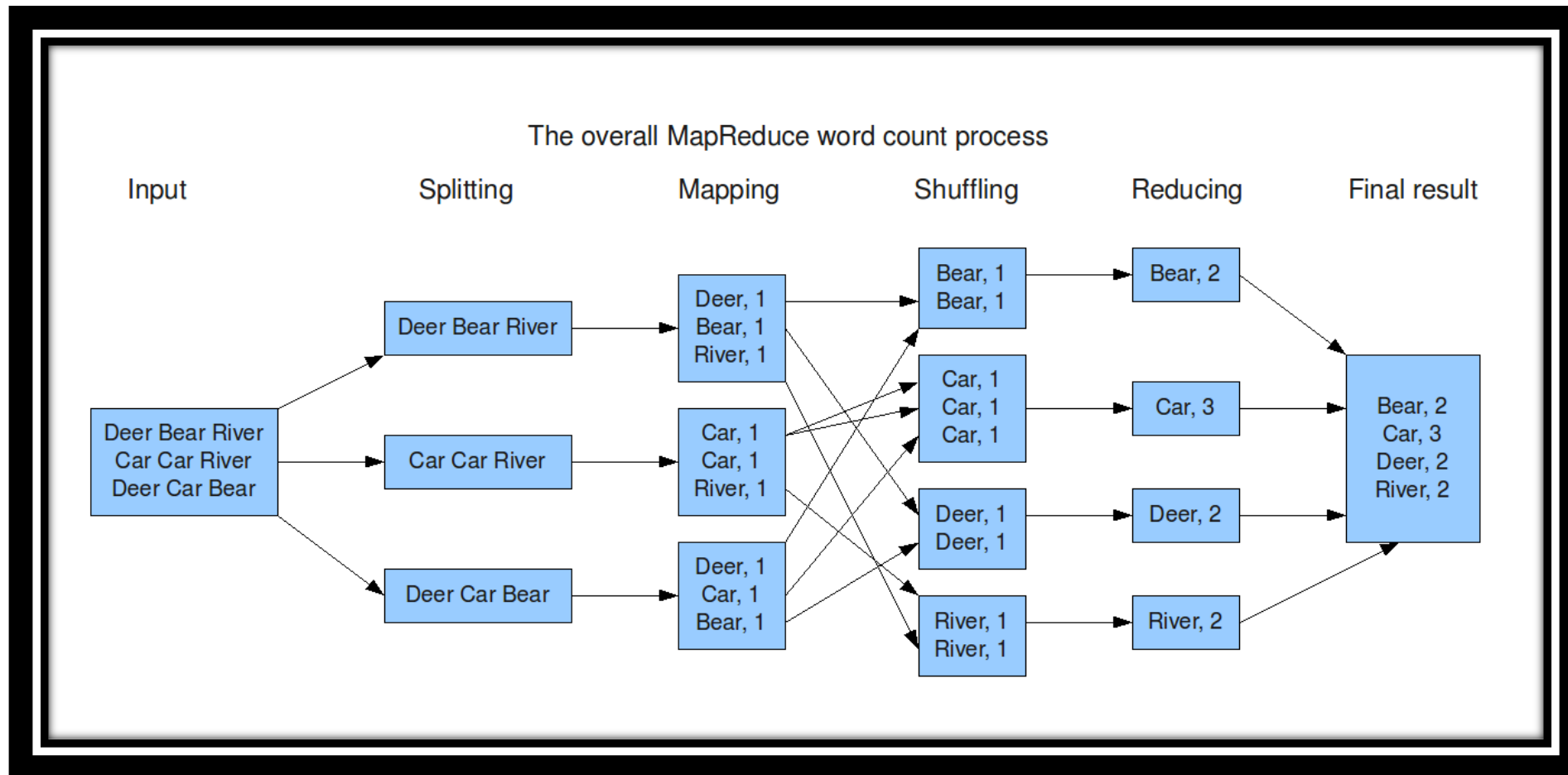
Shuffle and Sort

- Output from the mapper is sorted by key
- All values with the same key are guaranteed to go to the same machine

The Reducer

- Called once for each unique key
- Gets a list of all values associated with a key as input
- The reducer outputs zero or more final key/value pairs
 - Usually just one output per input key

MapReduce: Word Count



like three people
count together

Hadoop Architecture

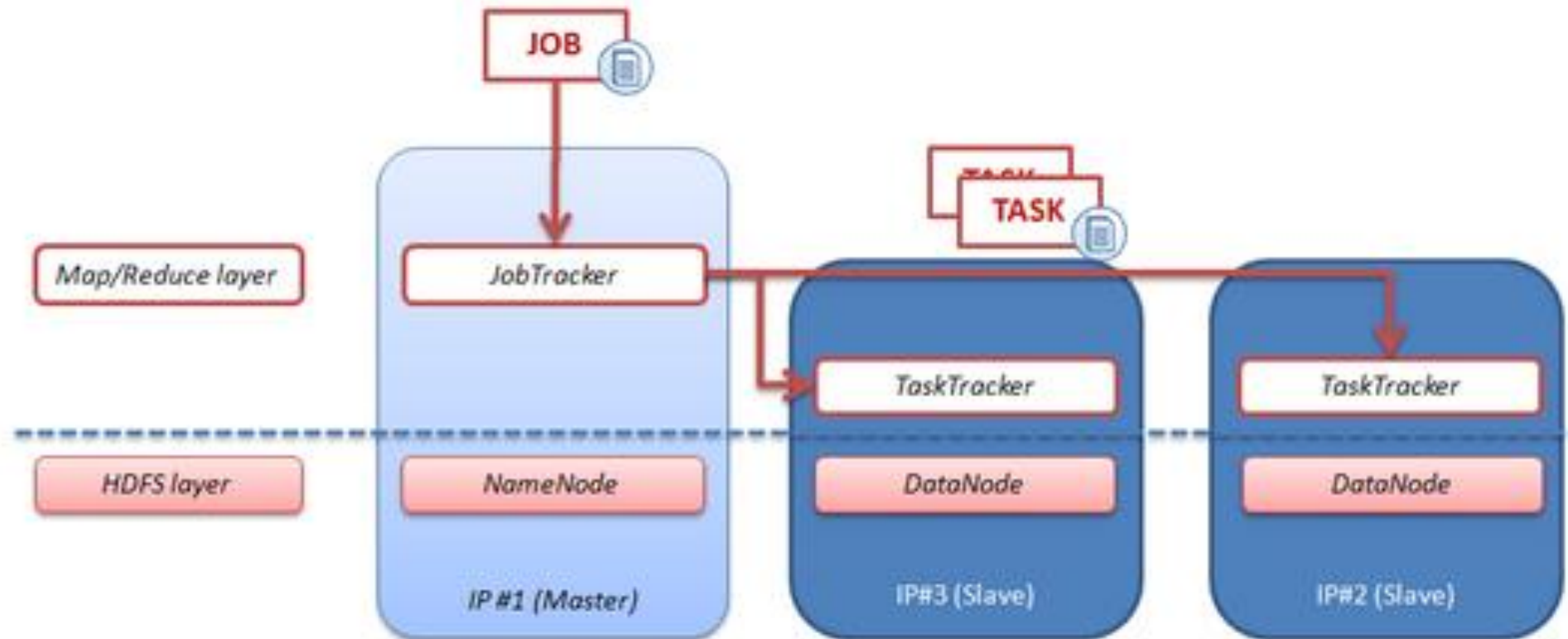
What parts actually make up a Hadoop cluster

- NameNode
 - Holds the metadata for the HDFS
- Secondary NameNode
 - Performs housekeeping functions for the NameNode
- DataNode
 - Stores the actual HDFS data blocks
- JobTracker
 - Manages MapReduce jobs
- TaskTracker
 - Monitors individual Map and Reduce tasks

JobTracker and TaskTracker

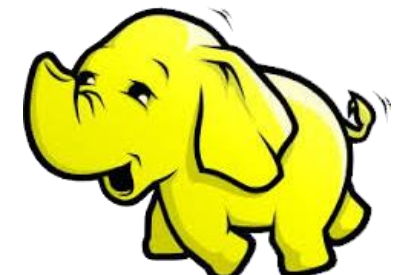
- JobTracker
 - Determines the execution plan for the job
 - Assigns individual tasks
- TaskTracker
 - Keeps track of the performance of an individual mapper or reducer

High-level architecture of Hadoop



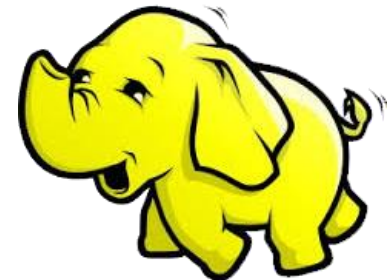
Hadoop's Architecture

- Hadoop Distributed Filesystem
- Tailored to needs of MapReduce
- Targeted towards many reads of filestreams
- Writes are more costly
- High degree of data replication (3x by default)
- No need for RAID on normal nodes
- Large blocksize (64MB)
- Location awareness of DataNodes in network



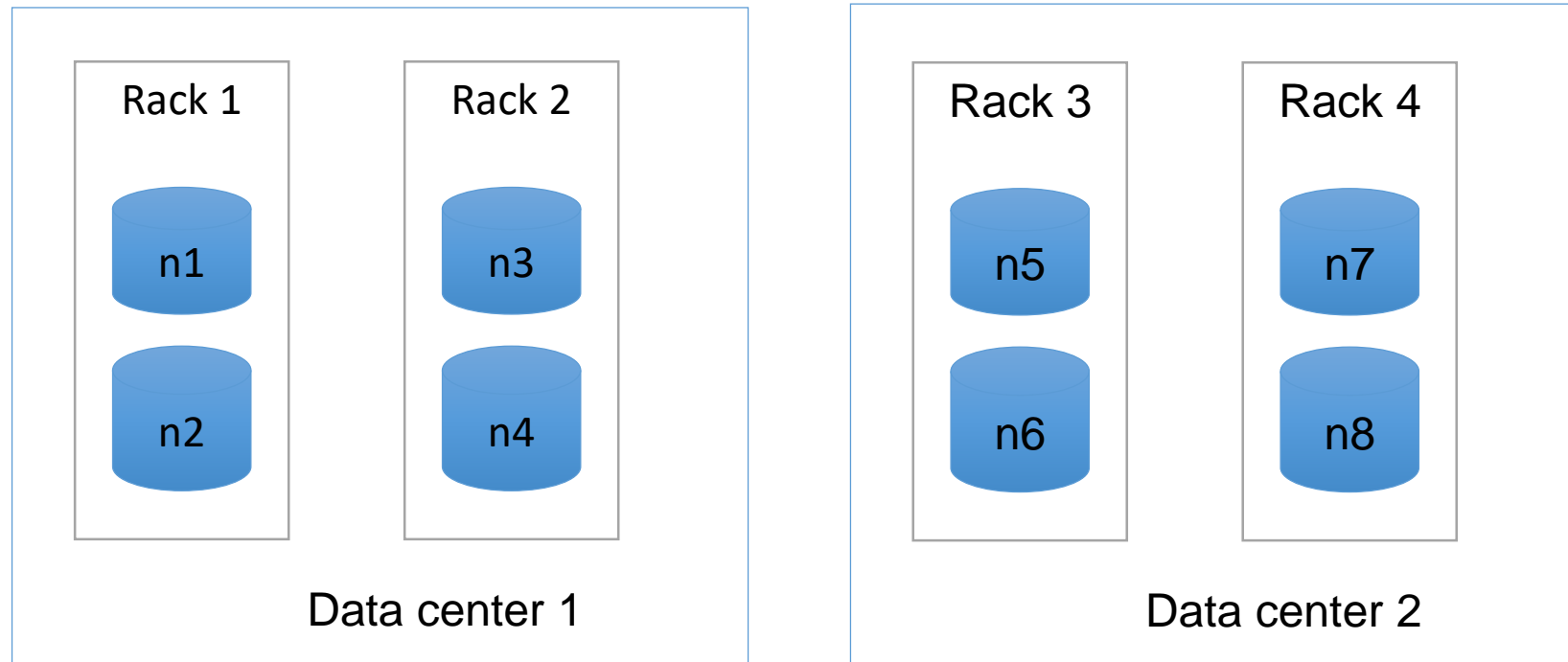
Hadoop's Architecture

- Distributed, with some centralization
- Main nodes of cluster are where most of the computational power and storage of the system lies
- Main nodes run **TaskTracker** to accept and reply to MapReduce tasks, and also **DataNode** to store needed blocks closely as possible
- Central control node runs **NameNode** to keep track of HDFS directories & files, and **JobTracker** to dispatch compute tasks to TaskTracker
- Written in Java, also supports Python and Ruby



HDFS network topology

- HDFS takes a simple approach:
 - ❑ See the network as a tree
 - ❑ Distance between two nodes is the sum of their distances to their closest common ancestor



HDFS filesystem commands

1. List the contents of a directory
 - `hadoop fs -ls`
2. Create a directory in HDFS at given path(s)
 - `hadoop fs -mkdir <directory name>`
3. Upload and download a file in HDFS
 - Upload: `hadoop fs -put <local file> <remote path>`
 - Download: `hadoop fs -get <file in HDFS> <local path>`
4. See contents of a file
 - `hadoop fs -cat <filename>`
5. Delete a file/directory in HDFS
 - `hadoop fs -rm/rmr <file or directory>`

HDFS filesystem commands

6. Move file from source to destination
 - `hadoop fs -mv <src> <dst>`
7. Report the amount of space used and availability
 - `hadoop fs -df hdfs:/`
8. How much space a directory occupies
 - `hadoop fs -du -s -h <dir name>`
9. Change permission of files
 - `sudo hadoop fs -chmod 600 <file>`
10. Change owner and group of files
 - `sudo hadoop fs -chown root:root <file>`

*HDFS admin commands

- DFSAdmin command

- -report: reports basic statistics of HDFS
- -safemode: though usually not required, an administrator can manually enter or leave safemode
 - enter, leave, get, wait
- -refreshNodes: updates the set of hosts allowed to connect to namenode

Usage: `hadoop dfsadmin [-report] [-safemode enter | leave | get | wait] [-refreshNodes] [-finalizeUpgrade] [-upgradeProgress status | details | force] [-metasave filename] [-setQuota <quota> <dirname>...<dirname>] [-clrQuota <dirname>...<dirname>] [-help [cmd]]`

Installation and configuration

Next week: Lab 3: Cloudera Hadoop Installation

Questions?