HADOOP ECOSYSTEM

CKME 134 – BIG DATA ANALYTICS TOOLS

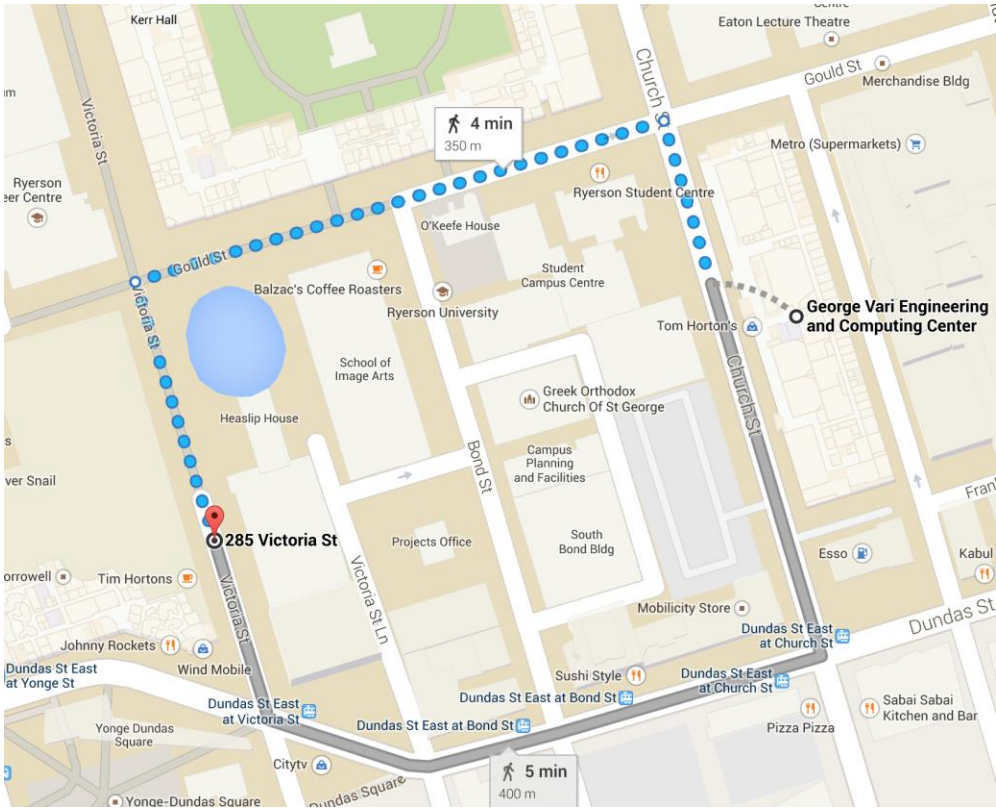
RYERSON UNIVERSITY

SPRING 2015

Instructor: Shaohua Zhang

General Course Information

- Instructor
  - Shaohua Zhang
  - Ryerson shaohua.zhang@ryerson.ca
  - Personal shaohua.zhang@live.com
- GA
  - Behjat Soltanifar
  - behjat.soltanifar@ryerson.ca
- Lectures
  - 6:30~8:30
  - ENGLG06
- Lab
  - 8:30~9:30
  - 285 Victoria St (403/404)
    - Take the elevator to 4FL



Course Outline (*subject to change*)

1. Intro to Big Data
2. Distributed Computing and MapReduce
3. Hadoop Ecosystem
4. Intro to Hive
5. Pig
6. Advanced Pig
7. Hadoop Performance Optimization
8. Big Data Use Cases: Location Intelligence and Marketing Analytics
9. Big Data Use Cases: Recommendation Engine and Computational Advertising
10. Hadoop In Action: Building Data Pipelines

Lecture 2 Recap

- Reading Materials
file://localhost/Users/DSinmotion/Dropbox/Startup/RyersonToolsCourse/Winter 2015/Session2-Distributed-computing-hadoop-and-mapreduce.pptx - 47. Recommended Readings - MapReduce

Lecture 2 Recap - Feedback From You

- Add Python Examples (will do! ☺)
- Add supplementary lab material (will do! ☺)
- The "More/less lab time" discussion…
  - We will get more hands-on starting Lecture 4 (Hive)
  - BUT…
    - Theory is important
    - Students come from different background and have various levels of skillsets
      - hard to balance!
- Practicing after class is more important! → will provide more supplementary materials

Postal code: M5S2J7
Cluster: 15

Lecture 2 Recap – Clustering

15 - Electric Avenues

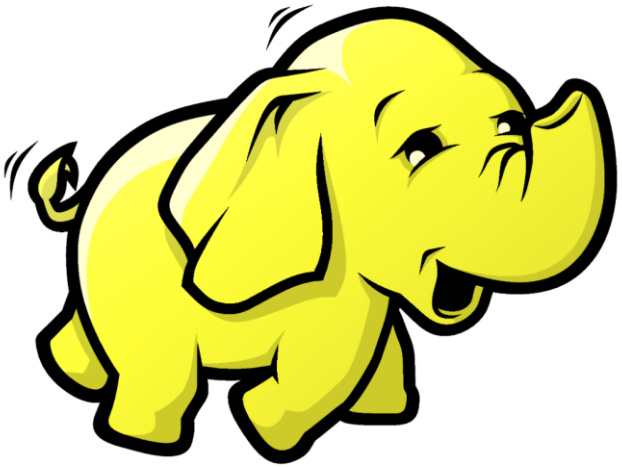
Social Group: U2 - Urban Young

Lifestage: Y1 - Singles Scene

Young, upper-middle-class urban singles and couples

Electric Avenues presents a classic portrait of young singles and couples pursuing lively urban lifestyles. Concentrated in Vancouver, Toronto, Ottawa-Gatineau and Calgary, their older, crowded neighbourhoods are known as havens for university graduates who rent apartments, have mid-level jobs and enjoy active leisure lives. While residents here have above-average household incomes, their spending power appears greater because so many households are childless. They spend freely on music, books, natural foods and electronics. They have high rates for going to bars, nightclubs and music festivals. Many engage in athletic activities such as jogging, baseball, canoeing and racquet sports. Progressive in their outlook—they support *Sexual Permissiveness* and the *Pursuit of Originality*—they like to acquire the latest in fashion, food and wine, often making their purchases online.

Hadoop



Disk Capacity

- While disk capacity increased, the cost has decreased significantly

Year	Capacity (GB)	Cost per GB (USD)
1997	2.1	$157
2004	200	$1.05
2012	3,000	$0.05

Disk Transfer Rate Challenge

- However, the disk transfer rate has not kept up with the pace

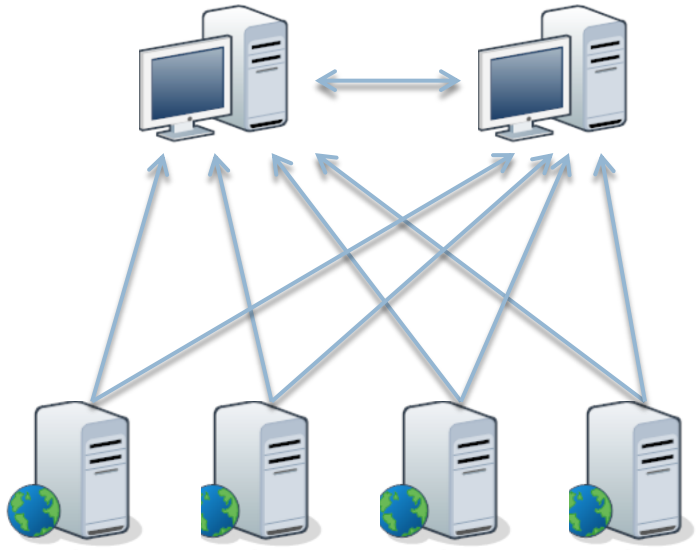
Year	Capacity (GB)	Transfer Rate (MB/s)	Disk Read Time
1997	2.1	16.6	126 seconds
2004	200	56.5	59 minutes
2012	3,000	210	3 hours, 58 minutes

Pitfalls of a Single Node Architecture

- Although we can process data faster today due to faster processor, accessing it is still slow
  - It takes 4 hours to read a 3TB disk
  - We cannot process the data until we have read it
- We've been building bigger, more powerful machines in the past few decades
  - but the single node architecture have two limitations
    - high cost
    - limited scalability

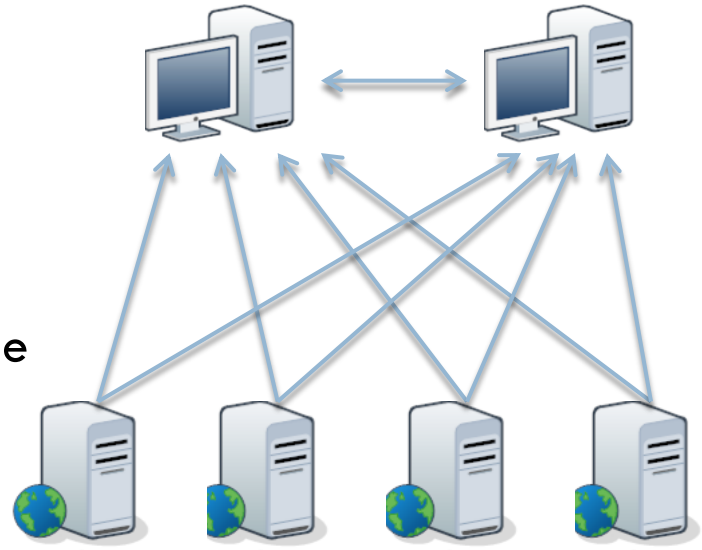
Traditional Distributed Systems

- Modern large scale processing is distributed across machines
  - Often hundreds or thousands of nodes
  - Common frameworks include MPI, PVM and Condor
- Focuses on distributing the processing workload
  - Powerful compute nodes
  - Separate systems for data storage
  - Fast network connections to connect them



Traditional Distributed Systems: Problems

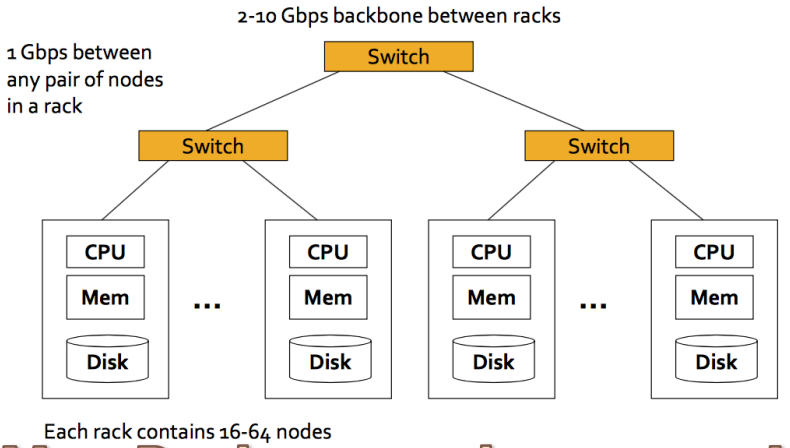
- Problems with these distributed systems:
  - Complex programming model
  - It is difficult to deal with partial failures of the system
  - Bandwidth limitations
  - Data consistency
  - Typically at compute time, data is copied to the compute nodes
    - This works fine with relatively small amounts of data, but doesn't scale to today's massive big data problem



Data Becomes the Bottleneck

- Traditional distributed systems don't scale to today's Internet-scale data
- Getting data to the computer processor becomes the bottleneck
  - Disk I/O is slow
  - Network bandwidth is bottleneck
- Solution → moving computation to the data!

• *Internet*	○ *2.5 exabytes (2.5×10^{18}) per day – 2012* ○ *2.3 zettabytes (2.3×10^{21}) per day - 2014*
• *Facebook*	○ *500+ terabytes per day* ○ *100+ petabytes in a single Hadoop cluster*



MapReduce to the rescue!

An Ideal Distributed System

- Handles failures well
  - (automatic) job should complete without manual intervention
  - (transparent) tasks assigned to a failed component are picked up by others
  - (graceful) failure only results in a proportional loss of load capacity
  - (recoverable) the capacity is reclaimed when the component is later replaced
  - (consistent) failure does not produce corruption or invalid results
- Scalability
  - Linear horizontal scalability (scale-out)
    - Adding new nodes should increase capacity proportionally
    - Shared nothing architecture
    - At a reasonable cost (commodity machines)
- Simple programming model
  - Programmers only focus on key functions while not worrying about distribution, parallelism, data transfer, failures etc.
  - Support many languages

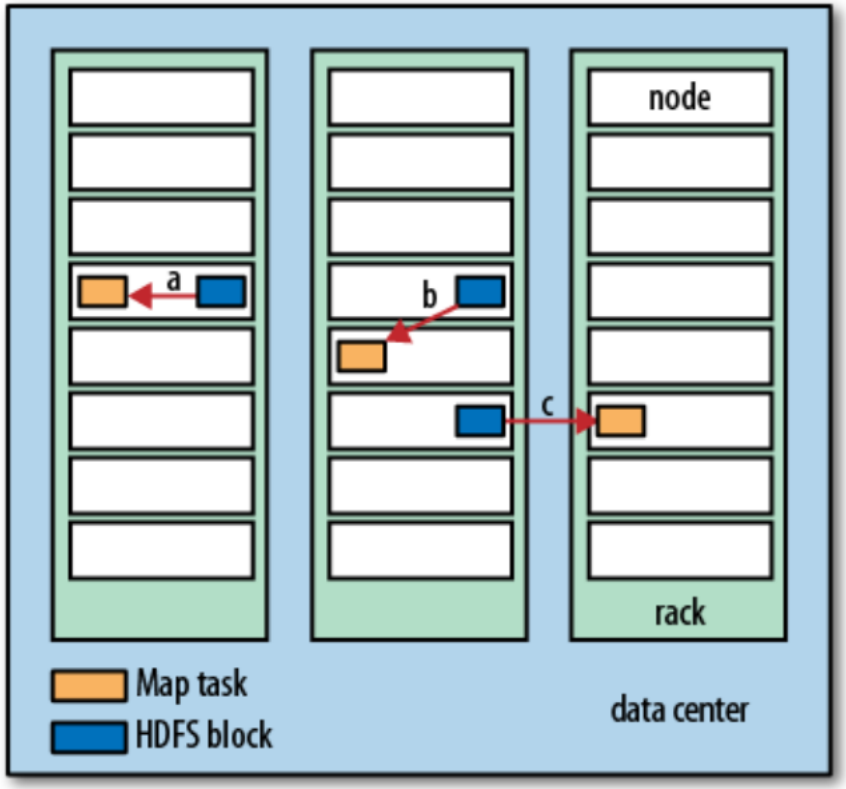
Disk Performance - Hadoop

Year	Capacity (GB)	Transfer Rate (MB/s)	Disk Read Time
1997	2.1	16.6	126 seconds
2004	200	56.5	59 minutes
2012	3,000	210	3 hours, 58 minutes

- Hadoop → Reading 1000 disks in parallel
  - 3TB in 15 seconds

Data Access - Hadoop

- Data Locality
  - Hadoop tries to process data on the same machine that stores it
  - This improves performance and conserves bandwidth
  - Brings computation to the data



Programming Model - Hadoop

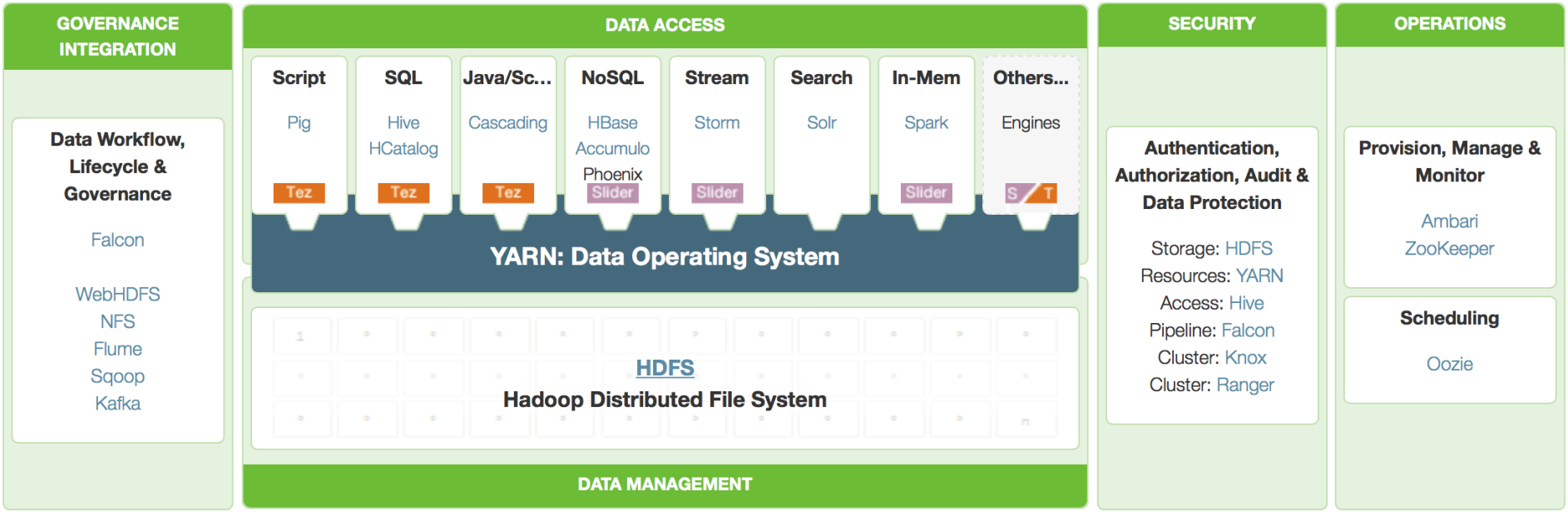
- MapReduce
  - Deals with one key value pair at a time
  - Complex details are abstracted away
    - No I/O
    - No networking code
    - No synchronization

Fault Tolerance - Hadoop

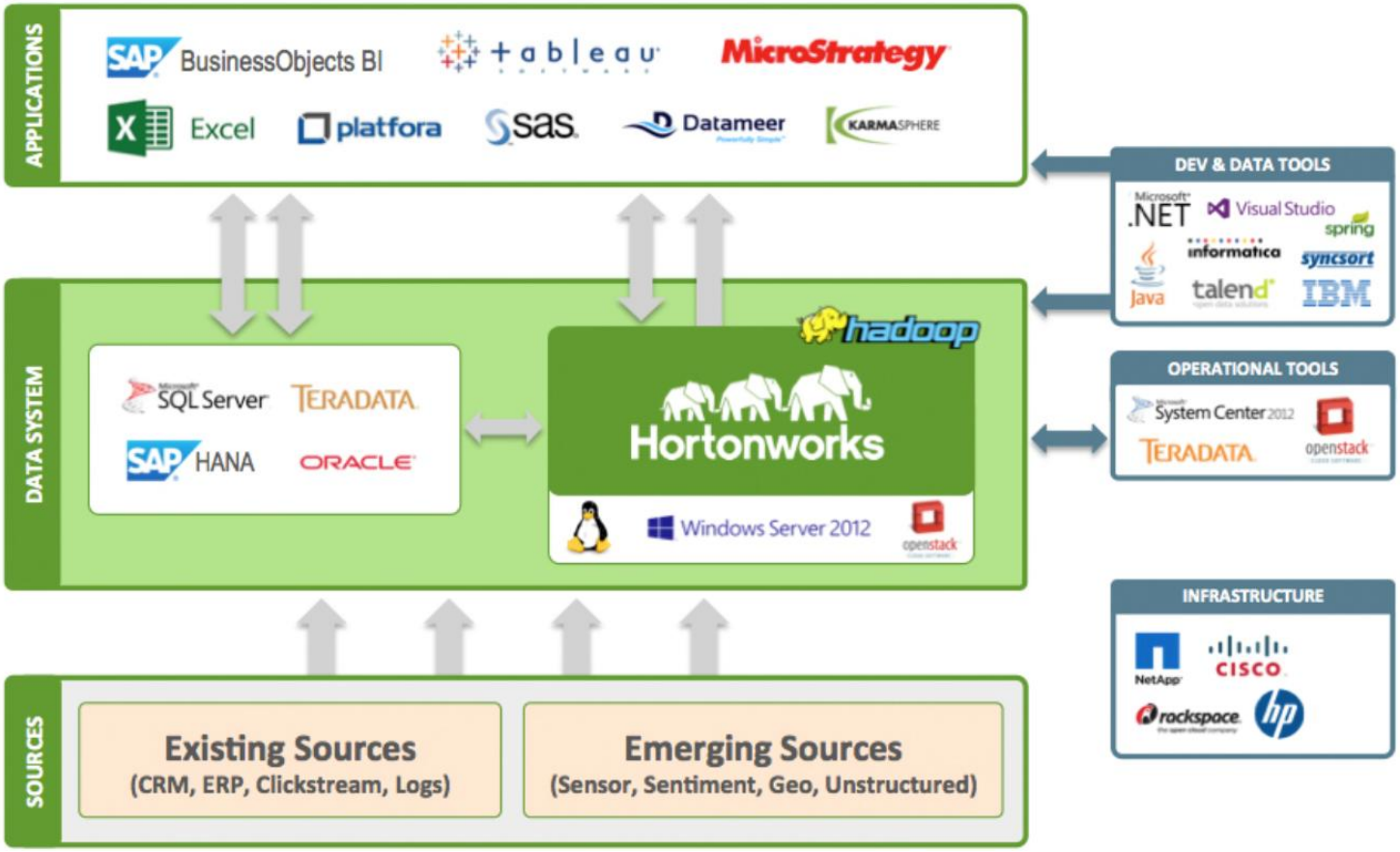
- If a node fails, the master will notice that failure and re-assign the task to a different node on the system
- Recovering a failed node doesn't affect nodes working on other portions of the data
- If a failed node restarts, it is automatically added back to the system and assigned new task
- If a node appears to run slowly, the master can redundantly execute another instance of the same task (speculative execution)

Hadoop Ecosystem

- Data analysis
  - Hive, Pig, Spark
- Machine Learning
  - Mahout, Spark (Mllib)
- Graph processing
  - Giraph, Spark (GraphX)
- Database Integration
  - Sqoop
- Scheduling & Workflow
  - Oozie
- Cluster management
  - Ambari
- Search
  - Solr
- NoSQL
  - Hbase, Cassandra
- Stream Processing
  - Storm



Enterprise Hadoop



Hadoop Core

- Hadoop is a system for large-scale data processing
  - Data storage: HDFS
  - Data processing: MapReduce

HDFS - Hadoop Distributed File System

HDFS

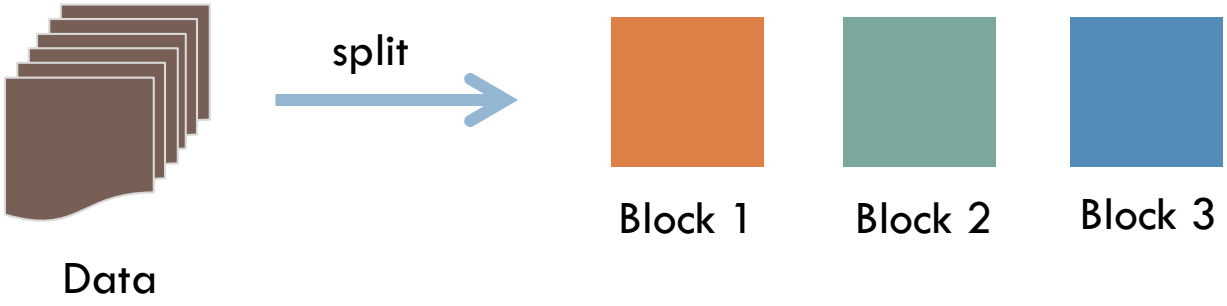
- A distributed file system that runs on large clusters of commodity machines
- Based on Google GFS paper
- Provides redundant storage for massive datasets

HDFS

- Advantage
  - Handling very large file
  - Streaming data access
  - Commodity hardware
  - Fault tolerance
  - Optimized for MapReduce programming
- Disadvantage
  - Low-latency data access (SQL, NoSQL)
  - Handling lots of small files

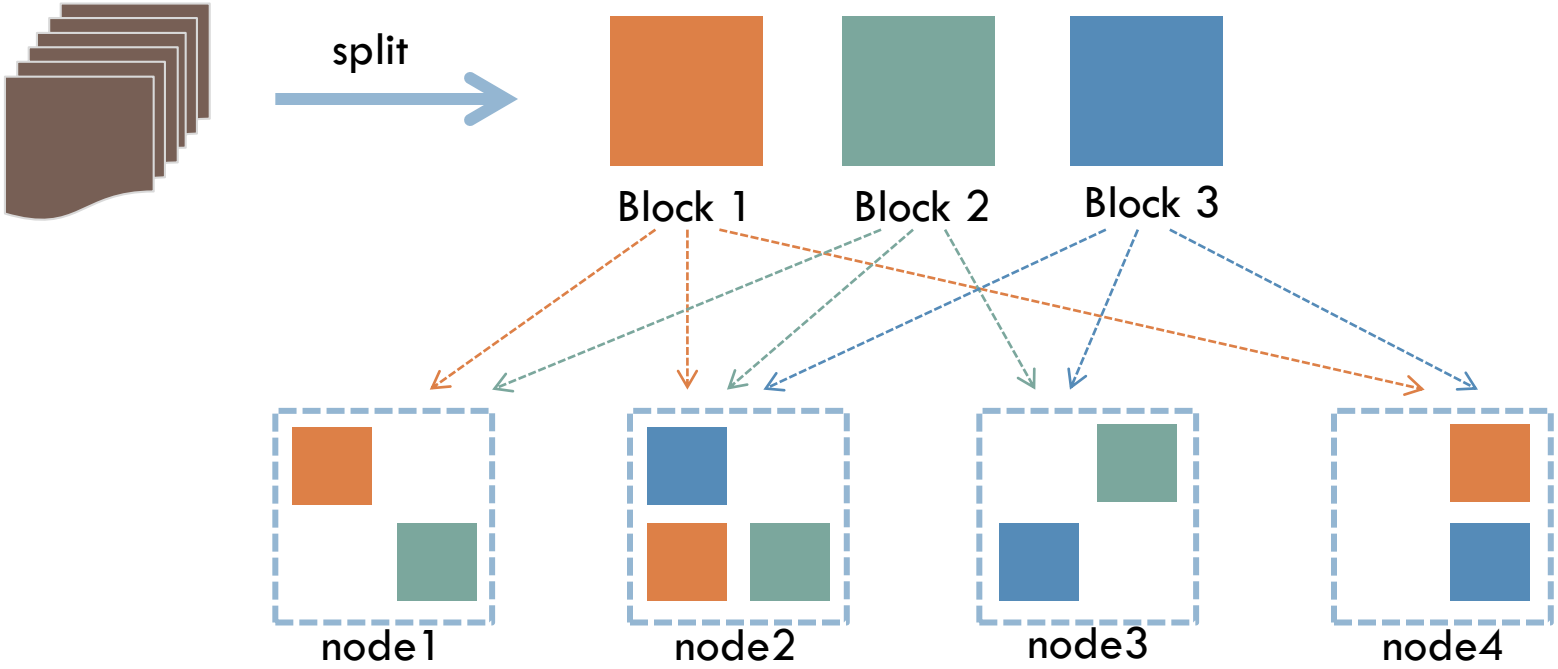
HDFS - Blocks

- When a file is added to HDFS, it is split into blocks
  - 64M by default,
  - Can be configured to 128M, 256M, 1G, etc.
- Why blocks?
  - Replication (fault tolerance)
  - Large file gets chunked and distributed easily
  - Data-local distributed computation (MapReduce)



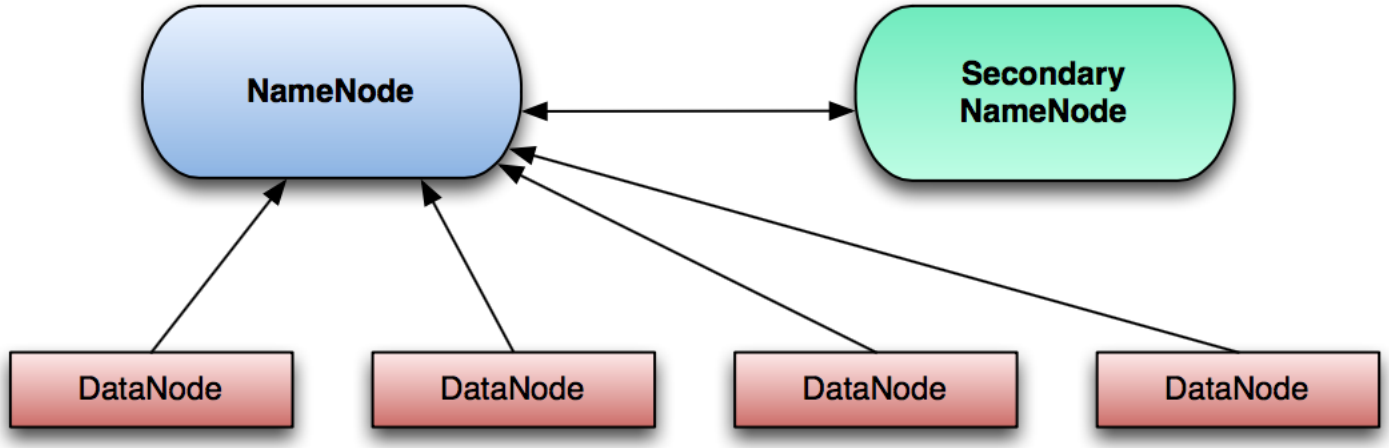
HDFS - Replication

- The blocks are replicated to nodes throughout the cluster
  - Based on the *replication factor* (3 by default)
- Replication increases reliability and performance
  - Reliability: can tolerate data loss
  - Performance: more opportunities for data locality



HDFS Architecture

- There're 3 daemons in "classical" HDFS
  - NameNode (master)
  - Secondary NameNode (master)
  - DataNode (slave)

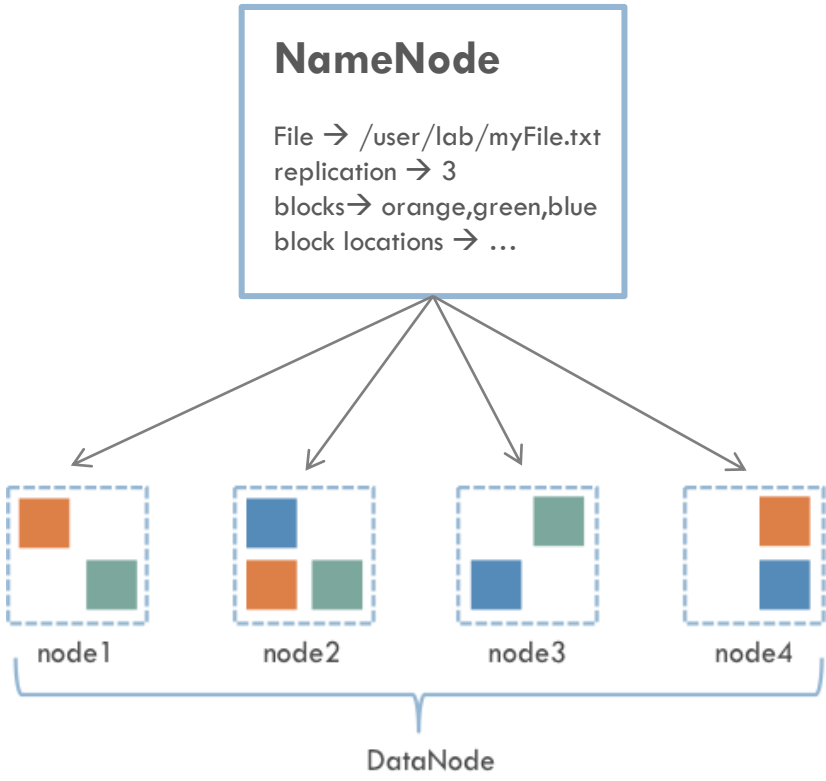


HDFS – NameNode and DataNode

- NameNode → master
  - Maintains filesystem tree and metadata in tree
  - Knows where all the blocks are stored for a file
- DataNode → worker
  - Store and retrieve data blocks
  - Report periodically with lists of blocks they stored

NameNode (master)

- The NameNode stores all metadata
  - Information about file locations in HDFS
  - Information about file ownership and permissions
  - Name of the individual blocks
  - Locations of the blocks
- Metadata is stored on disk and read into memory when the NameNode daemon starts up
- Changes/Edits to the files are written to the logs

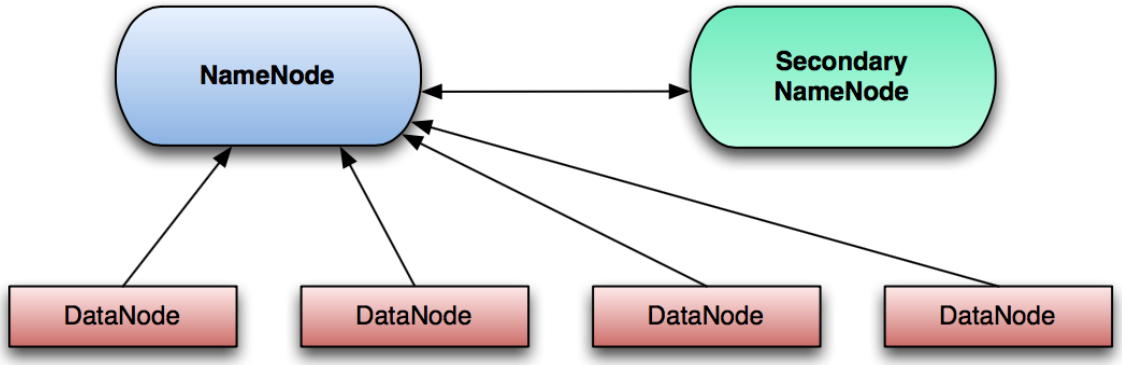


DataNode (slave)

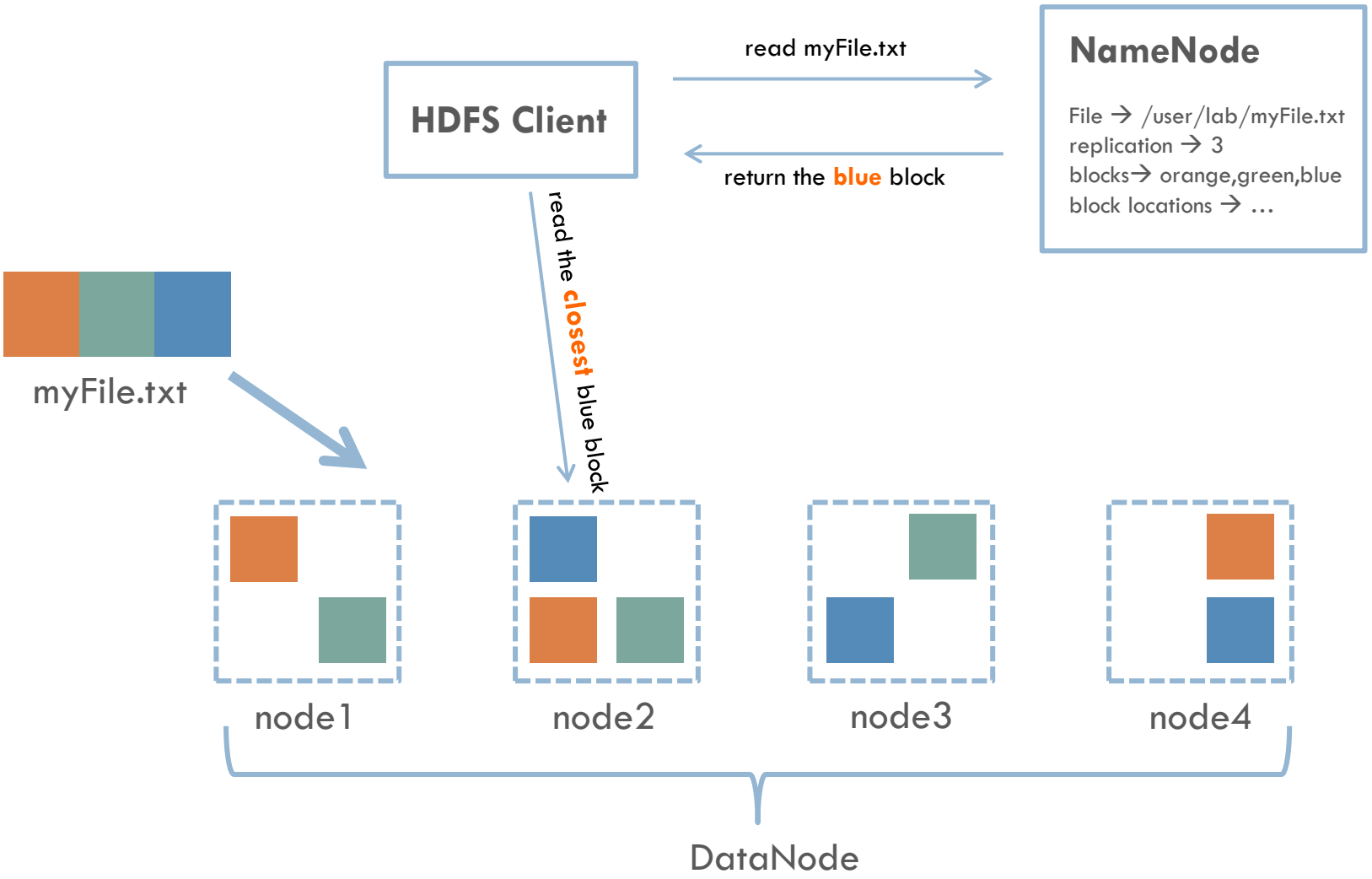
- Actual files/data are chunked into blocks and stored on the data nodes
  - Block name “blk_xxxxx” maps to a block name in the NameNode
  - The location of the blocks are stored in NameNode instead
- Each block is replicated to different nodes for redundancy
- The DataNode daemon controls access to the blocks and communicates with the NameNode

Secondary NameNode (master)

- The Secondary NameNode is not a backup for the NameNode
  - It provides memory-intensive administrative functions for the NameNode
    - Secondary NameNode periodically combines a prior snapshot of the file system metadata and edit logs into a new snapshot
    - It then transmits the new snapshot back to the NameNode

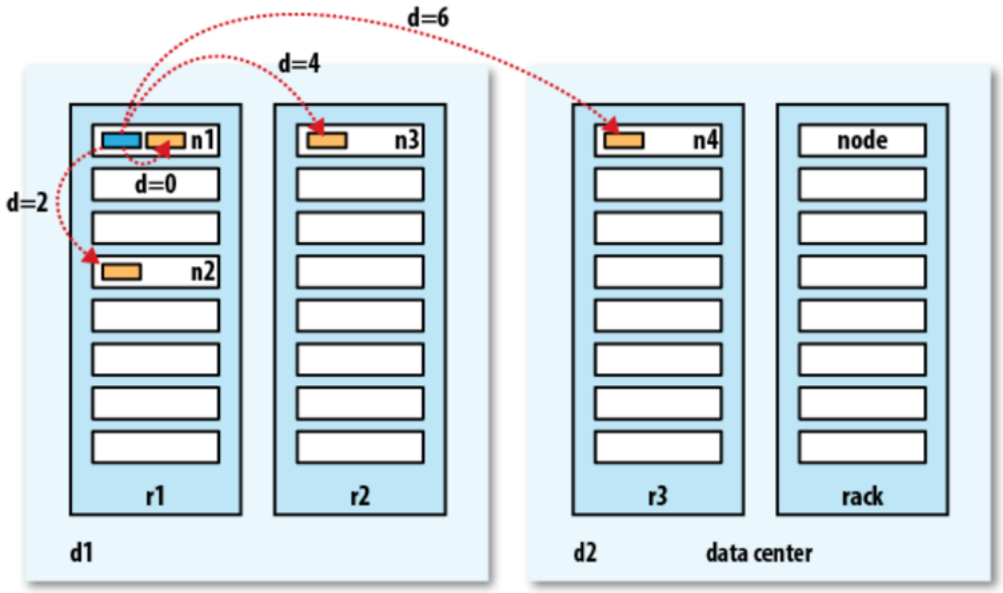


Anatomy of a File Read

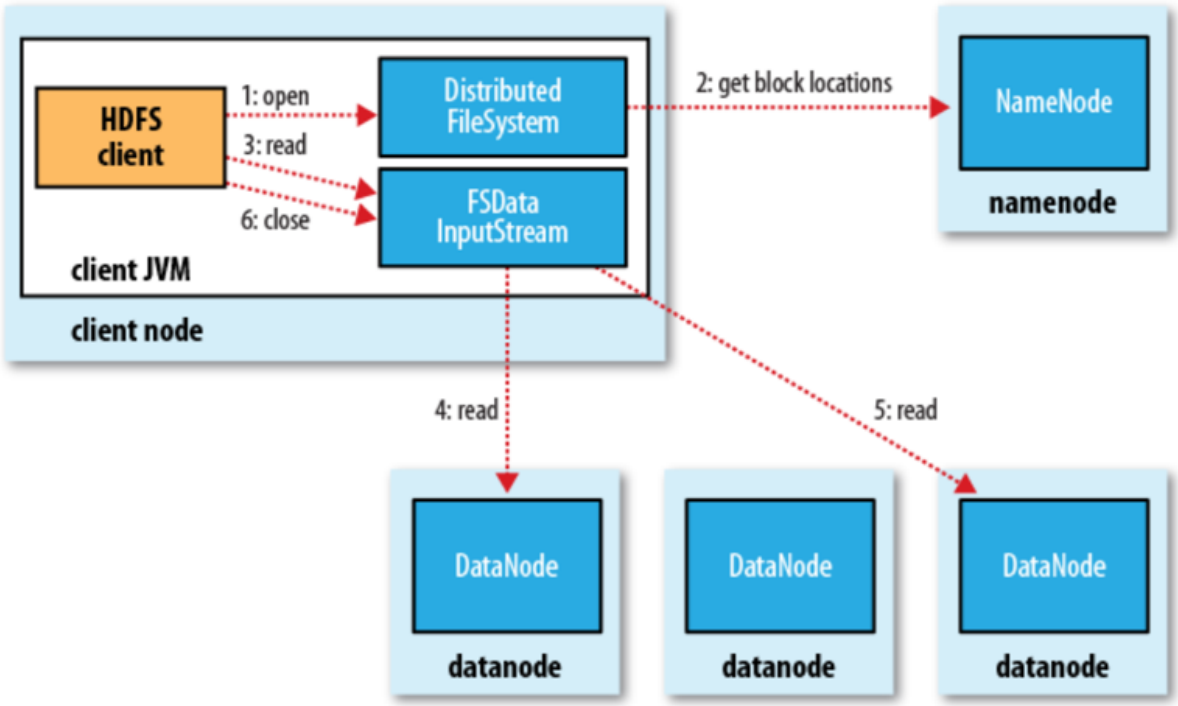


Network Distance in Hadoop

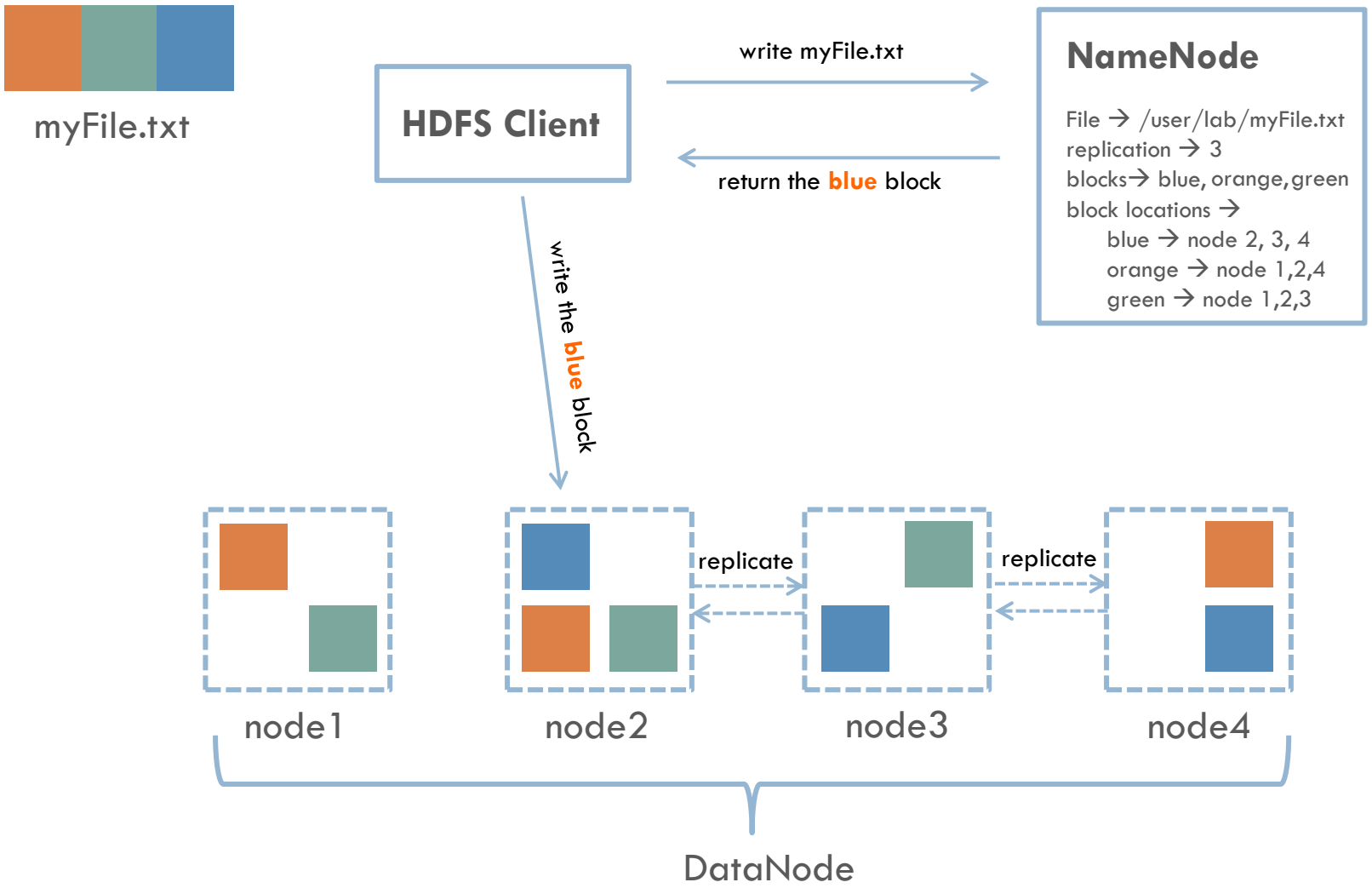
- How does Hadoop decide which block of data is closest?



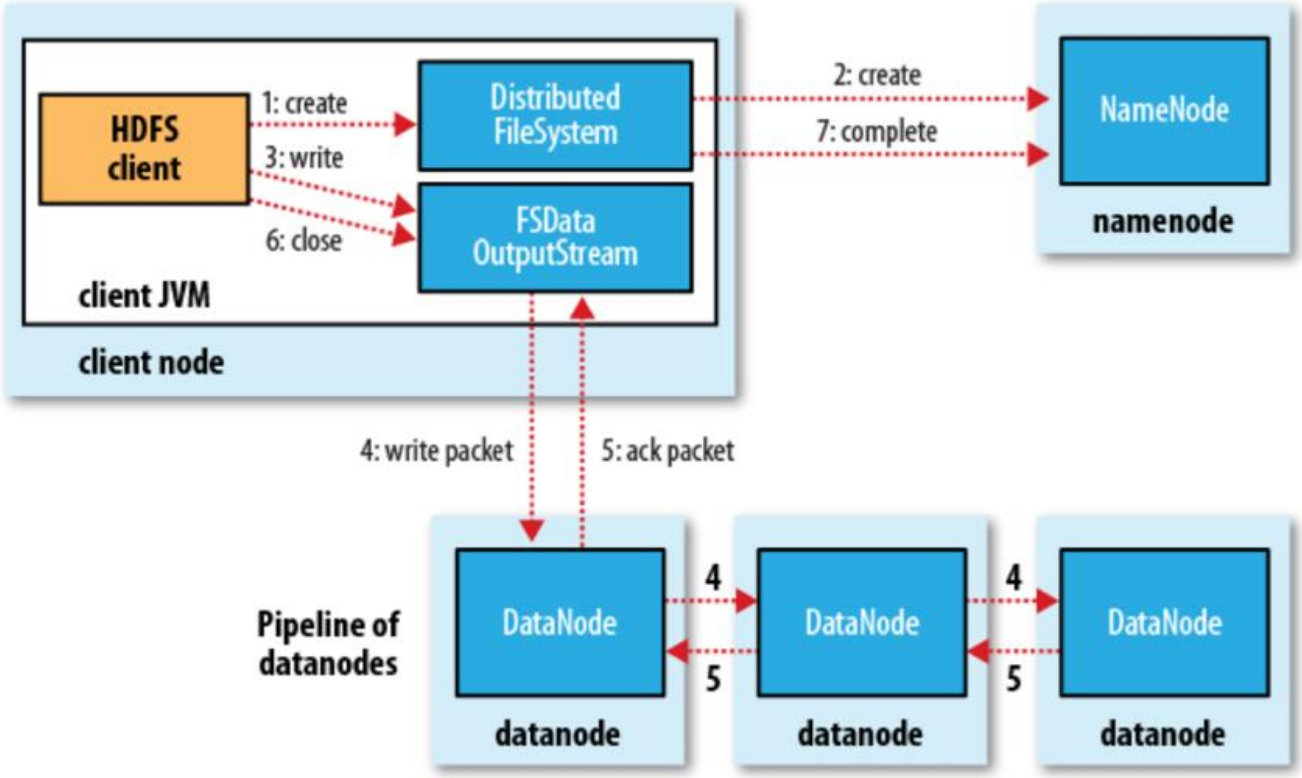
Anatomy of a File Read



Anatomy of a File Write



Anatomy of a File Write

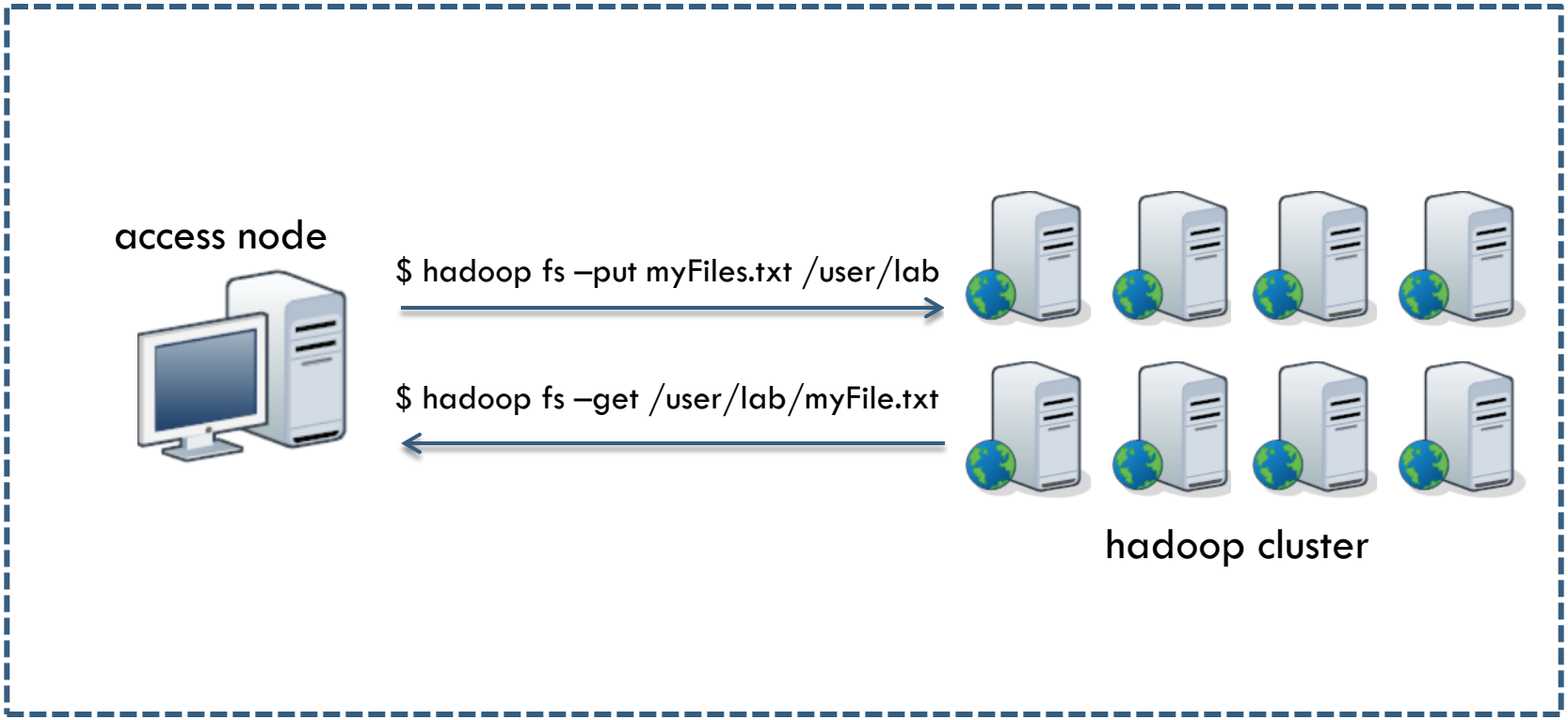


Block Replication Strategy

- Hadoop's default strategy is to place the first replica on the same node as the client
- The second replica is placed on a different rack from the first (off-rack), chosen at random.
- The third replica is placed on the same rack as the second, but on a different node chosen at random.

HDFS – CLI (command line)

- Users typically access HDFS via *hadoop fs* command



hadoop fs Command Examples

Command	*Comments*
hadoop fs –mkdir /user/lab	*create a directory on hdfs*
hadoop fs –ls /user/lab	*list the files in the directory*
hadoop fs –put myFile.txt /user/lab	*move a file from local fs to hdfs*
hadoop fs –cat /user/lab/myFile.txt | head	*display the content of a file*
hadoop fs –get /user/lab/myFile.txt	*move a file from hdfs to local fs*
hadoop fs –rmdir /user/lab	*remove a directory on hdfs*

DEMO TIME!

shakespeare data: http://www.gutenberg.org/files/100/100.txt

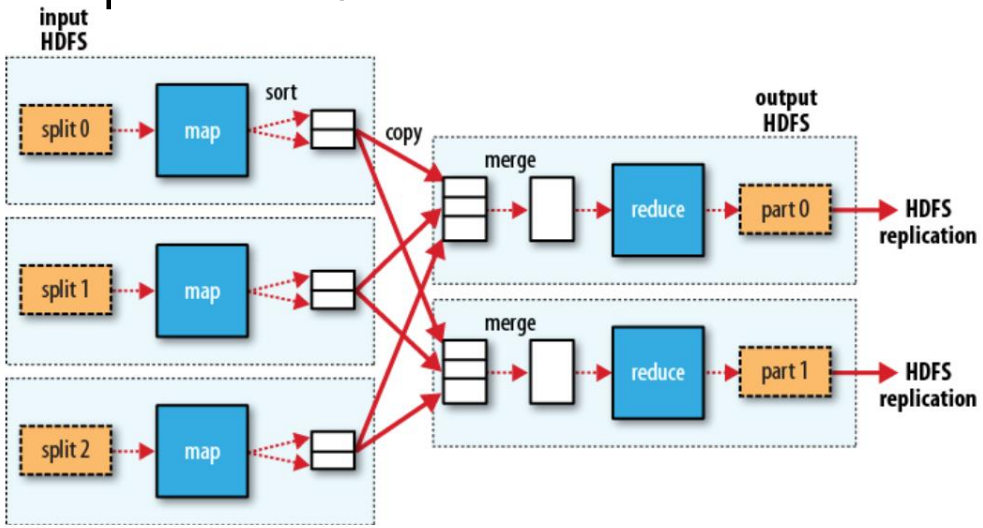
MapReduce

MapReduce

- *MapReduce is a computing model that decomposes large data manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers*
- Each node processes data stored on that node
- Consists of two phases
  - Map
  - Reduce

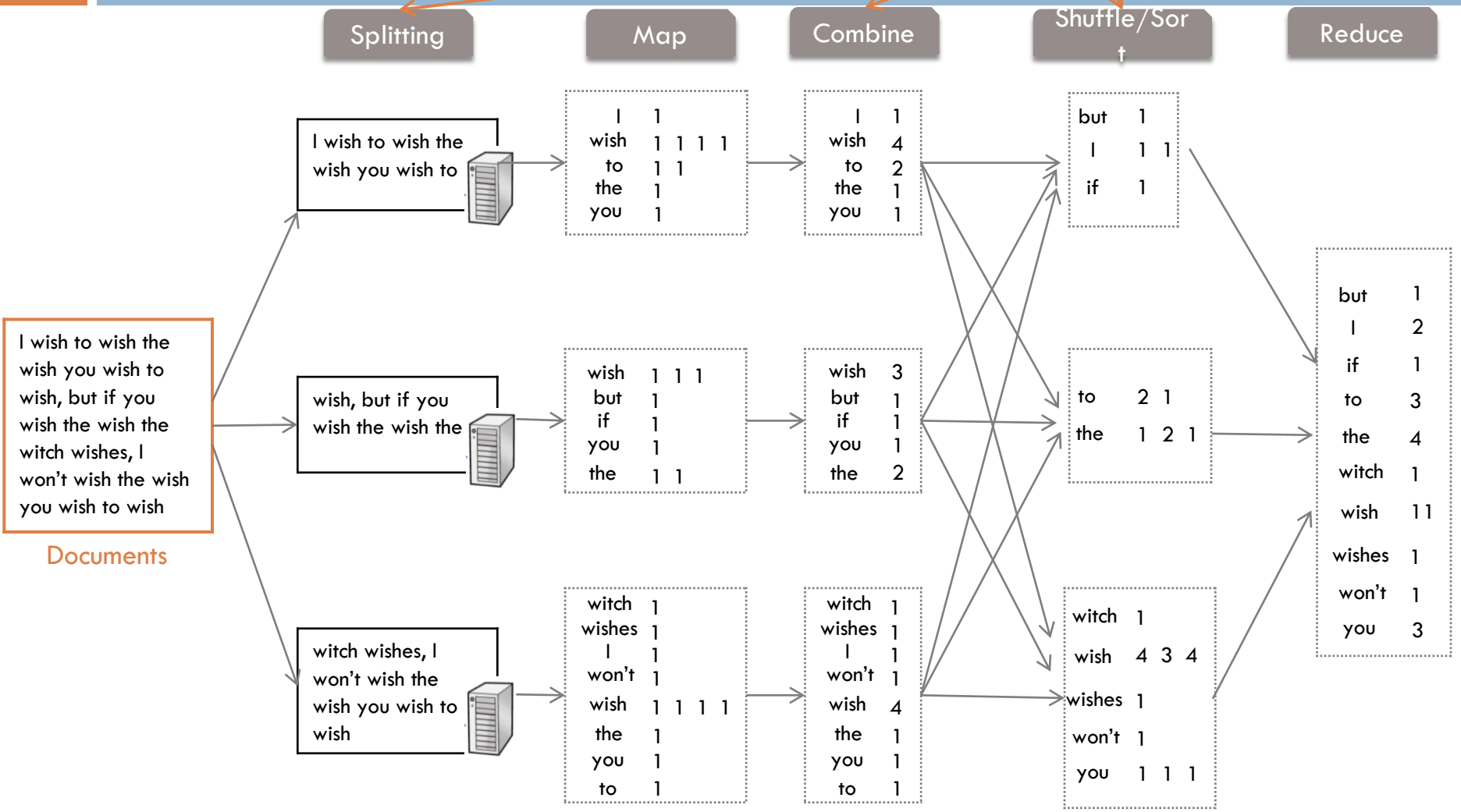
MapReduce – Map & Reduce

- Each mapper processes a single input split from HDFS
- Each map task process data one record at a time
- Each record has a *key* and a *value (key-value pair)*
- Shuffle and sort phase makes sure that all the values associated with the same intermediate key are sent to the same reducer
- Reducer receives the key and associated list of values and then does the reduce operations
- Reducer writes output to HDFS



Word Count – MapReduce

MapReduce handles these automatically for you!!

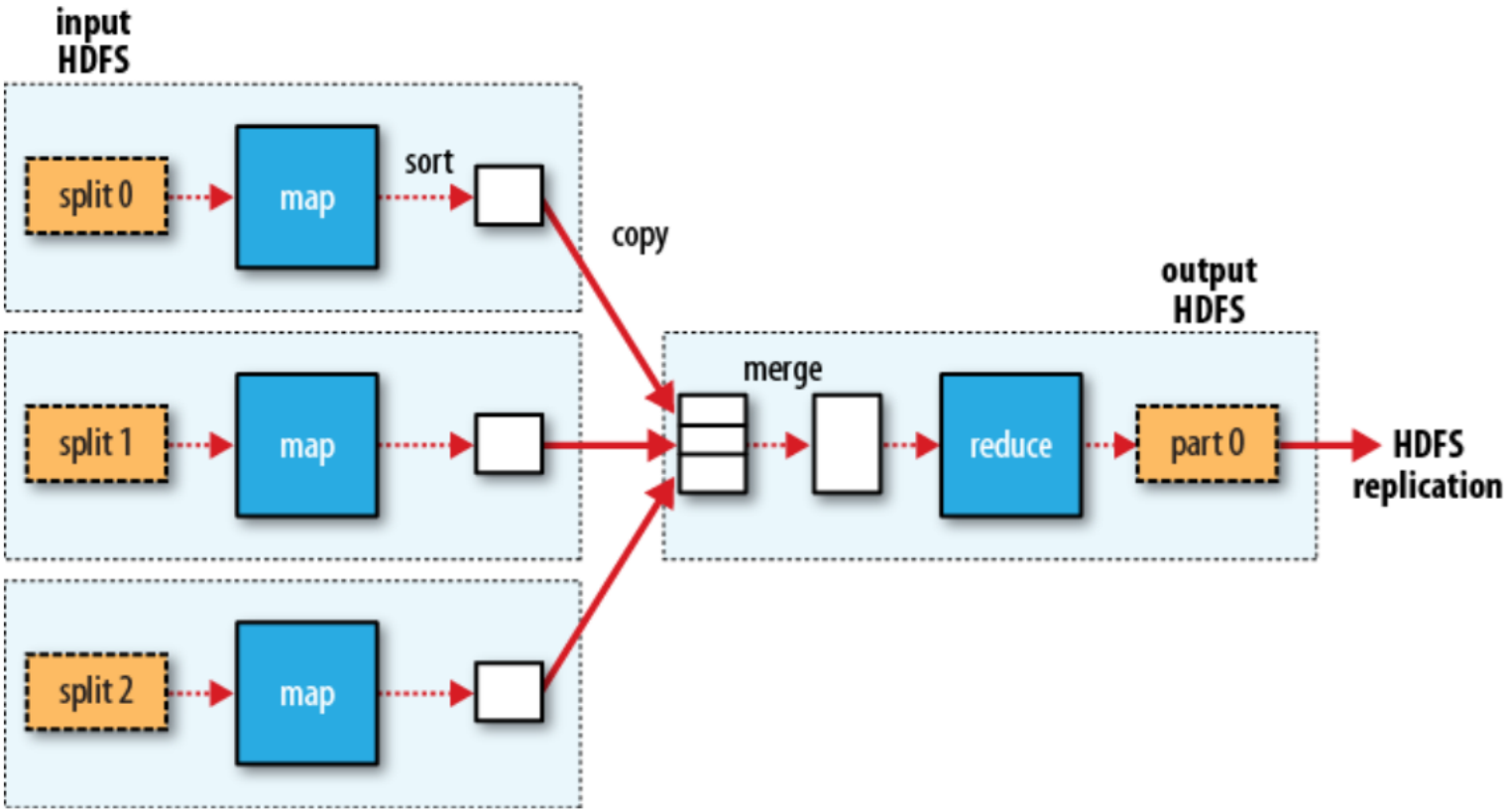


WordCount – MapReduce Demo

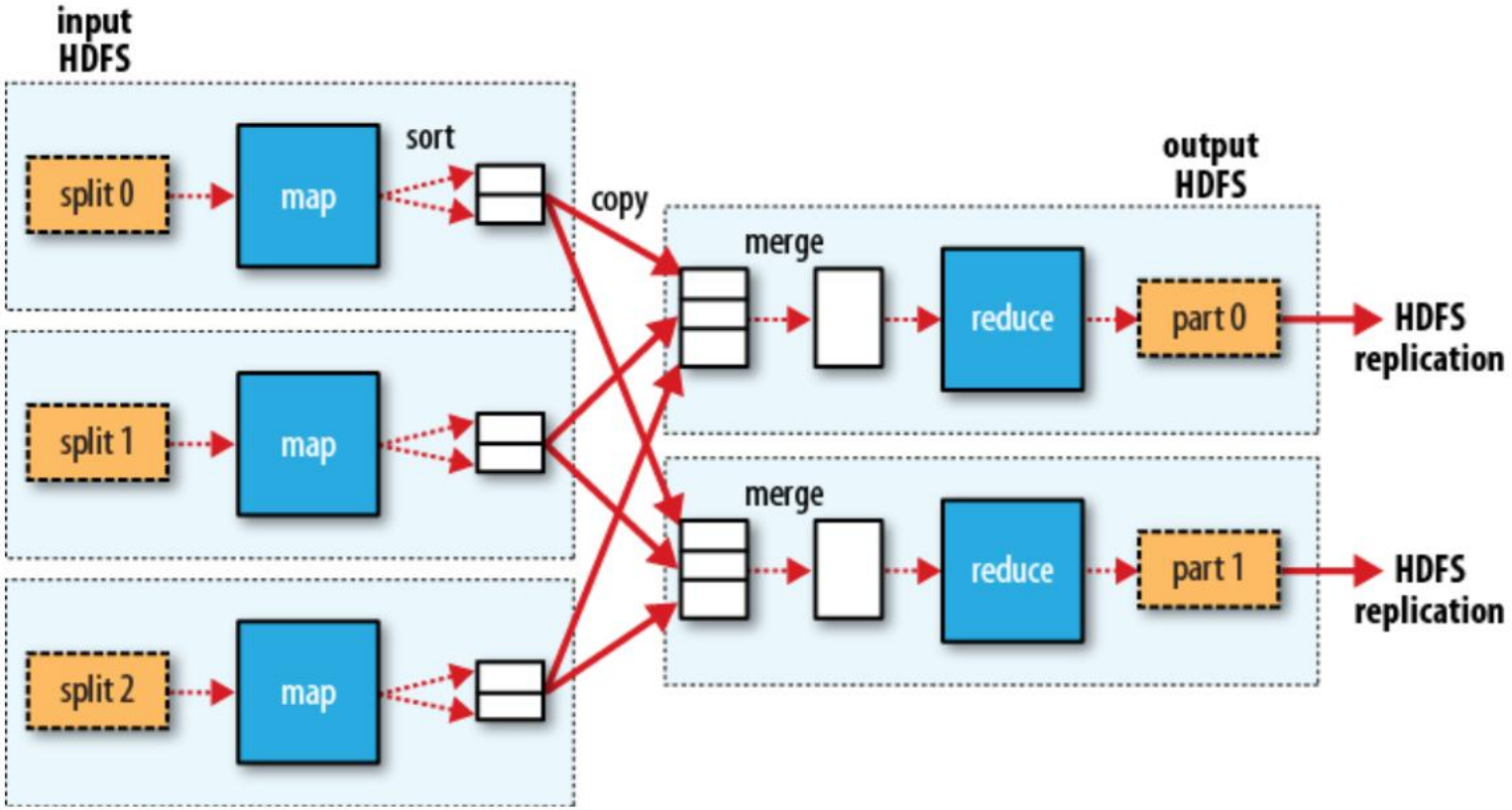
1. Download Shakespeare file
2. Put the shakespeare file into HDFS
3. Find your mapreduce-example jar file
   - $ find /usr/local/hadoop* -name *mapreduce-example*
4. Run java M/R wordcount example
   - $ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar wordcount /user/shaohua/shakespeare /user/shaohua/shakespeare-wc-out

DEMO TIME!

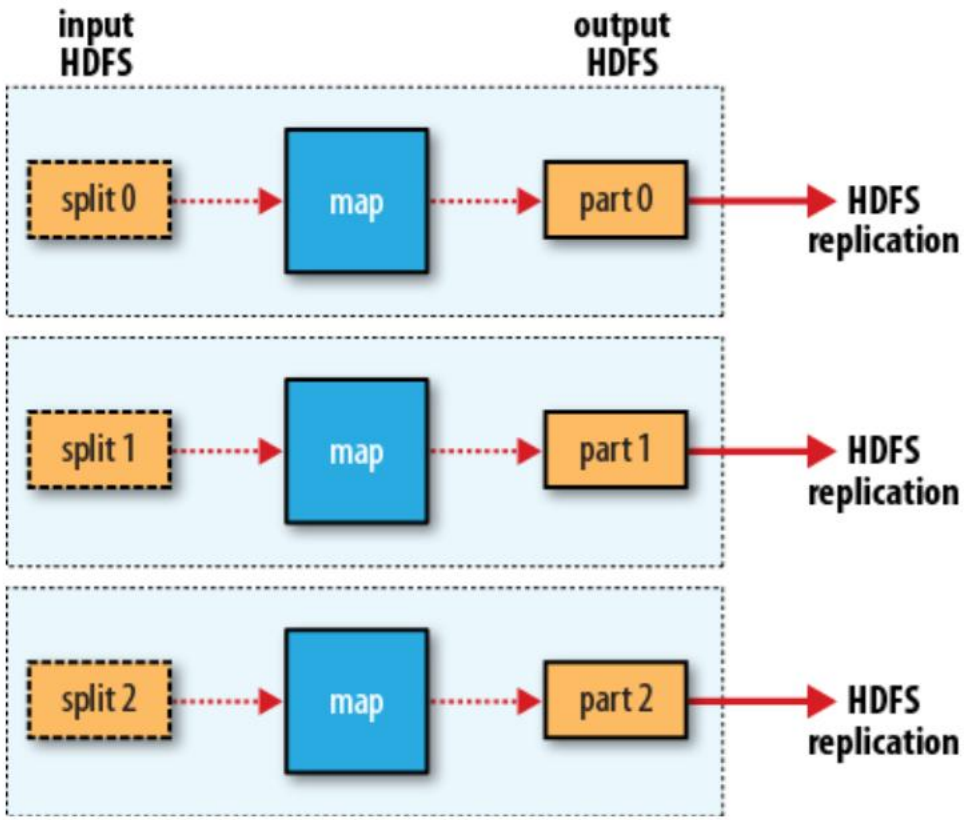
MapReduce – 3 mappers, 1 reducers



MapReduce – 3 mappers, 2 reducers



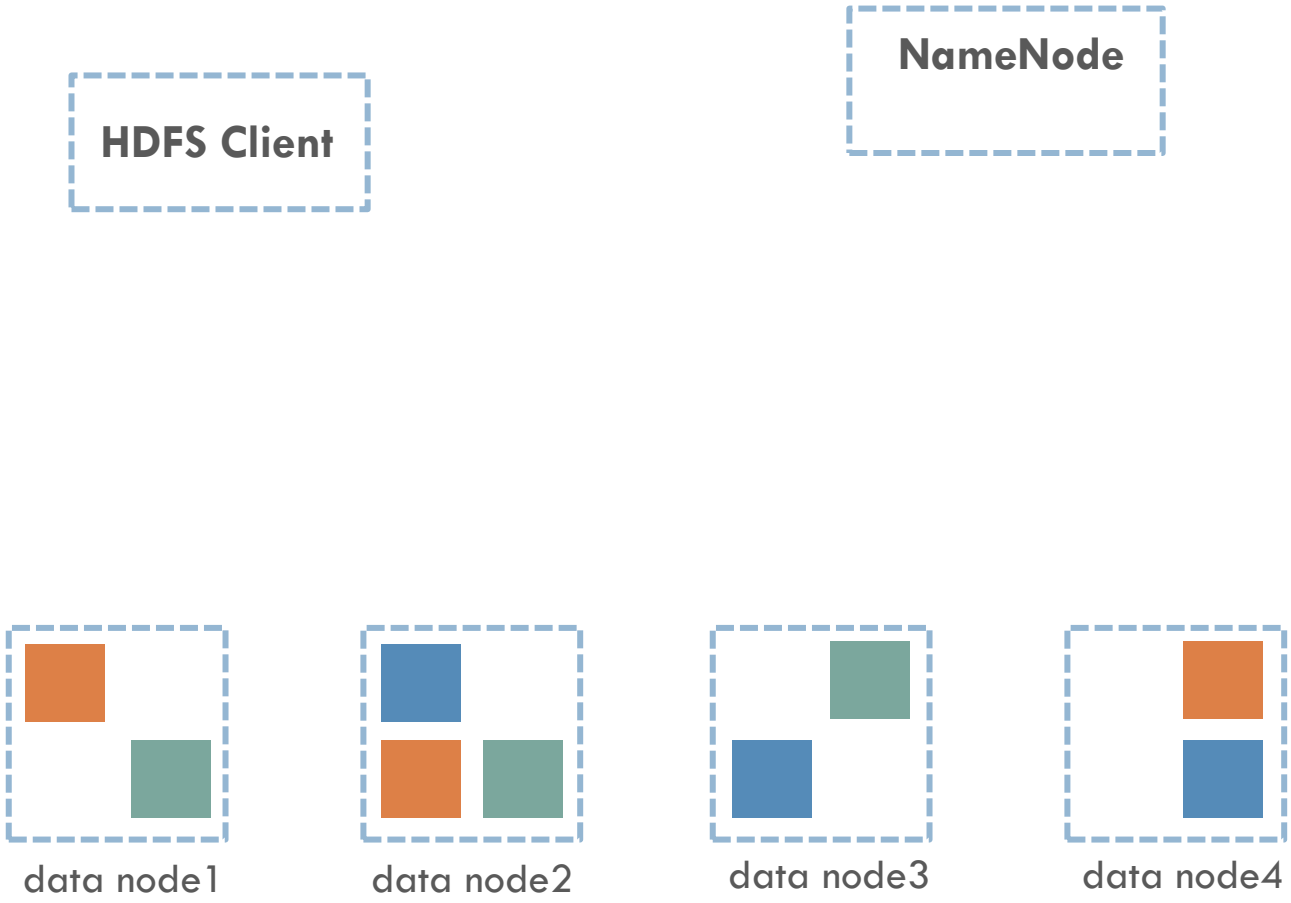
MapReduce – Map Only Job



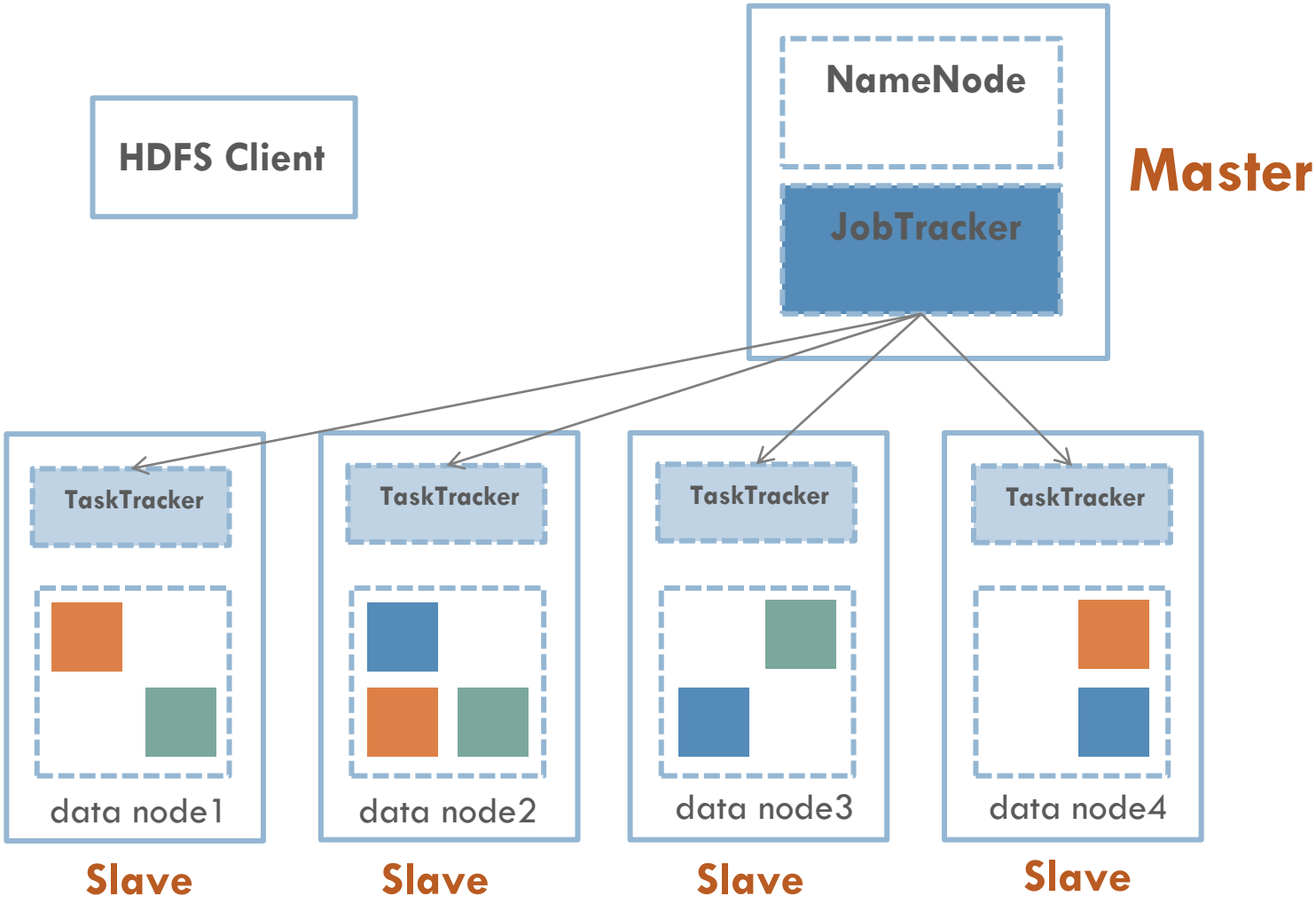
MapReduce Terminology

- Terminology
  - Job
    - Consists of a Mapper, a Reducer, and a list of input files
  - Task
    - An individual unit of work
    - A job is broken down to many tasks
      - map tasks or reduce tasks
  - Client
    - Machine on which the program runs

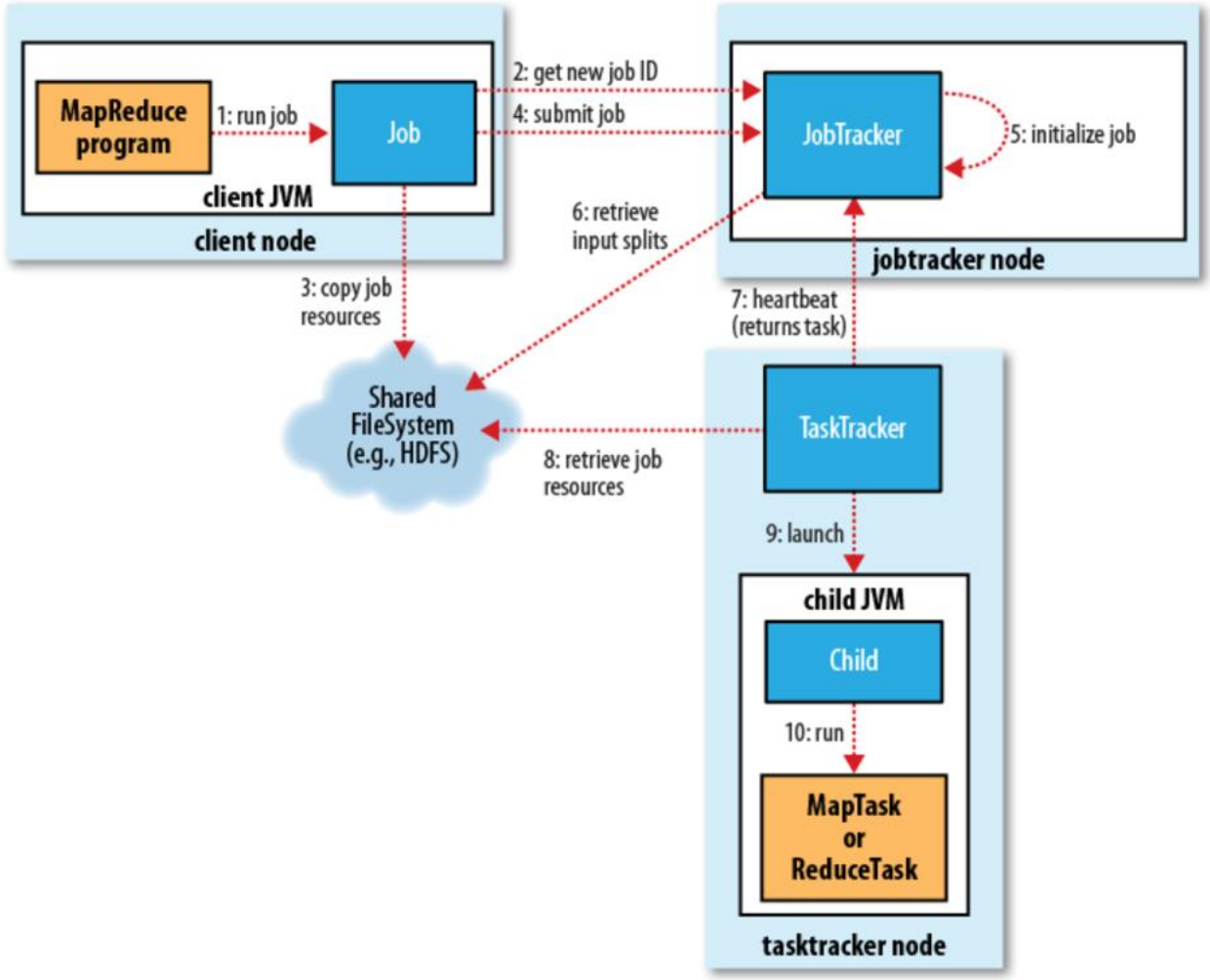
MapReduce Architecture



MapReduce Architecture



Anatomy of a MapReduce Job Run



Data Analysis - Hive

Apache Hive

- Hive is an abstraction on top of MapReduce
  - Developed at Facebook
- It allows users to query data in the Hadoop cluster without knowing Java and MapReduce
- Uses the HiveQL language
  - Very similar to SQL
- The Hive Interpreter runs on a client machine
  - Turns HiveQL queries into M/R jobs
  - Submits those jobs to the cluster
- Hive != SQL

Hive Query Example

```
CREATE TABLE docs (line STRING);

LOAD DATA INPATH 'docs' OVERWRITE INTO TABLE docs;

CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
  (SELECT explode(split(line, '\s')) AS word FROM docs) w
GROUP BY word
ORDER BY word;
```

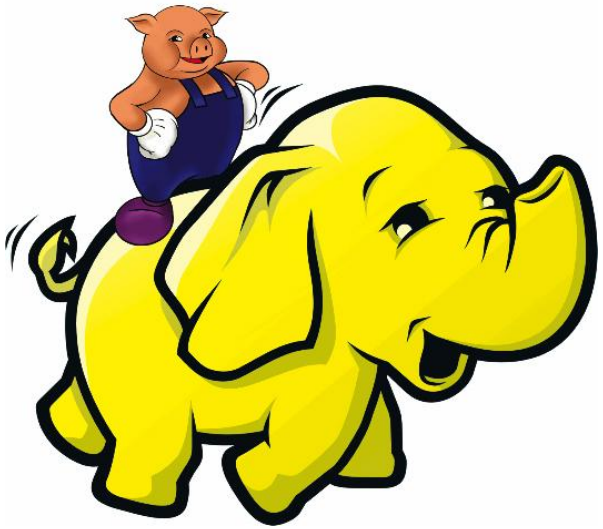
Hive vs. SQL

- Hive deals with unstructured data better
  - Word Count is not so easy in SQL
    - Hive deals with key-value pairs
    - SQL deals with rows/columns
- Hive is more scalable (Hadoop)
- Hive has dynamic schema
  - SQL has static schema
- Hive Limitations
  - No support for Update or Delete
  - No support for inserting single rows
  - Limited number of Built in functions
  - Not all Standard SQL is supported

Data Analysis - Pig

Apache Pig

- Pig Latin, a high level data processing language.
- An engine that executes Pig Latin locally or on a Hadoop cluster



Pig Latin Example

Query : Get the list of web pages visited by users whose age is between 20 and 29 years.

```
USERS = load 'users' as (uid, age);

USERS_20s = filter USERS by age >= 20 and age <= 29;

PVs = load 'pages' as (url, uid, timestamp);

PVs_u20s = join USERS_20s by uid, PVs by uid;
```

Why Pig?

- Faster development
  - Fewer lines of code
  - Don't re-invent the wheel
  - No M/R programming
- Flexible
  - Metadata is optional
  - Extensible (UDFs → Piggybank, DataFu, etc.)
  - Procedural programming

Pig Philosophy

- Pigs eats anything
  - Pig can operate on data whether it has metadata or not. It can operate on data that is relational, nested, or unstructured. And it can easily be extended to operate on data beyond files, including key/value stores, databases, etc.
- Pigs live anywhere
  - Pig is intended to be a language for parallel data processing. It is not tied to one particular parallel framework (e.g., Pig on Spark, Pig on Storm)
- Pigs are domestic animals
  - Designed to be easily controlled and modified by users via User-Defined-Functions (UDF) and Stream command, etc.
- Pigs fly

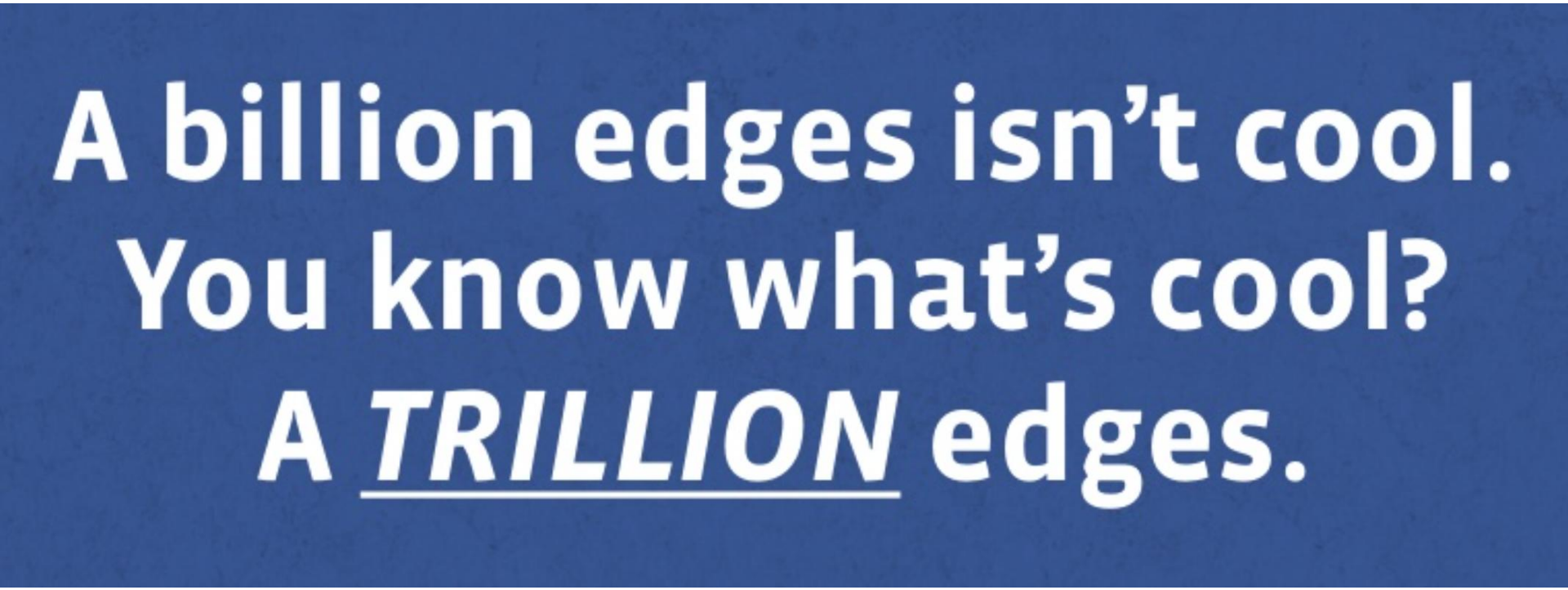
Apache Pig Philosophy: http://pig.apache.org/philosophy.html

Machine Learning - Mahout

Apache Mahout

- Mahout
  - Scalable machine learning libraries
  - (Mostly) Hadoop-based
- Algorithms
  - Clustering
    - K-Means/Canopy
    - Latent Dirichlet Allocation (LDA) → Topic Modeling
  - Classification
    - Logistic Regression
    - Naïve Bayes
    - Random Forest
  - Recommender Engine
    - User and Item-based recommenders
    - Matrix factorization based recommenders
- Moving to Spark and integrating H^2O

Graph Processing - Giraph


A billion edges isn't cool.
You know what's cool?
A TRILLION edges.

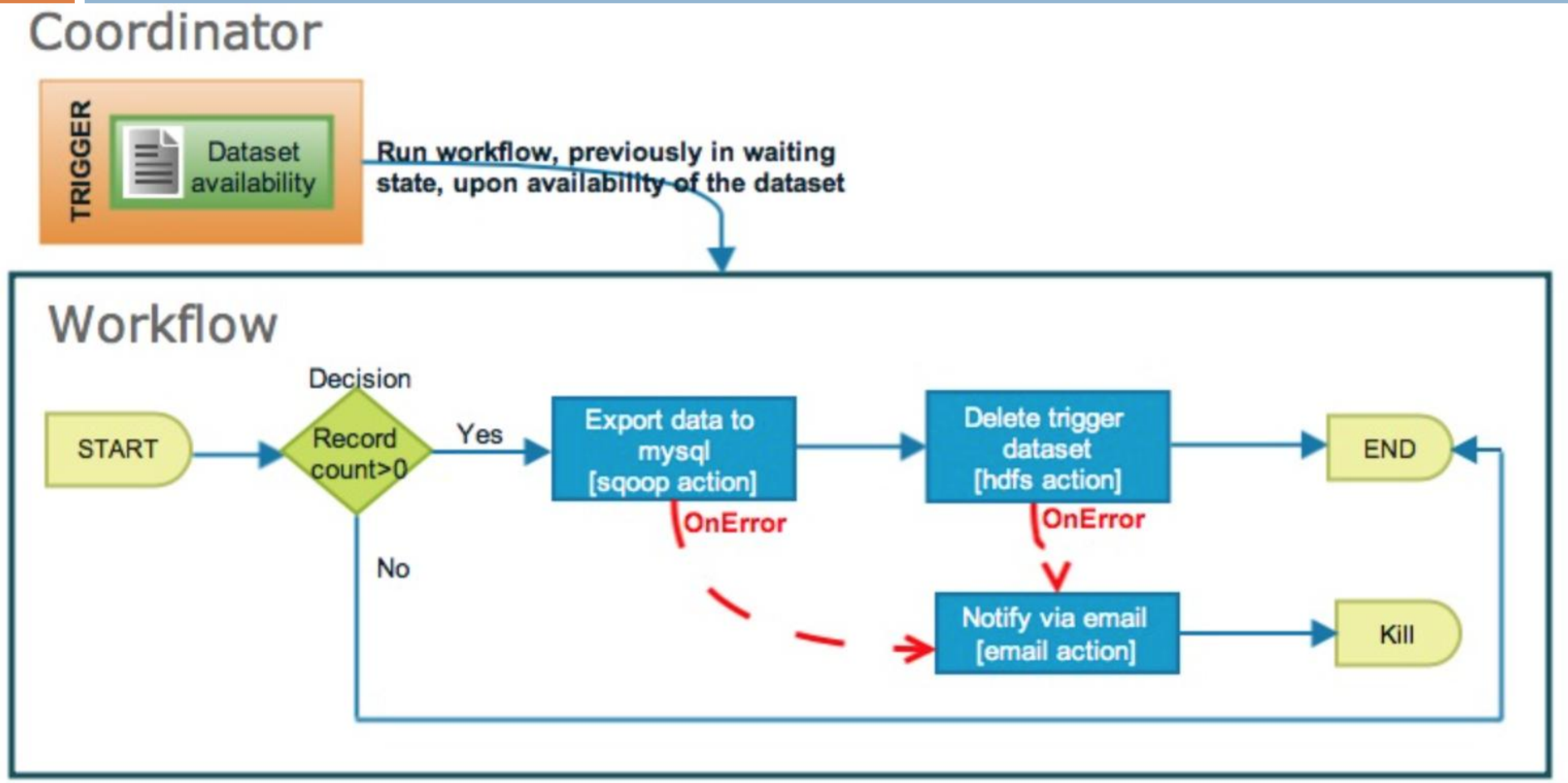

Page rank on 200 machines
with 1 trillion
(1,000,000,000,000) edges
<4 minutes / iteration!

Workflow Management - Oozie

Oozie

- A service for running and scheduling workflows of Hadoop jobs (including MapReduce, Pig, Hive, and Sqoop jobs)
- Oozie **Workflow** jobs are Directed Acyclical Graphs (DAGs) of actions.
- Oozie **Coordinator** jobs are recurrent Oozie Workflow jobs triggered by time (frequency) and data availabilty.

Workflow DAG

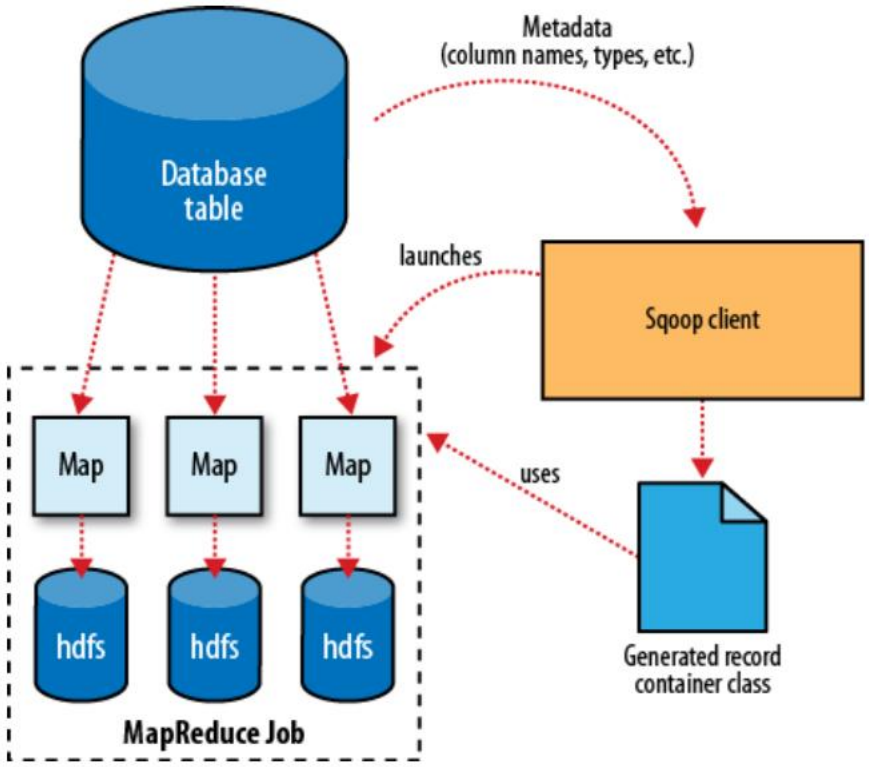


Database Integration - Sqoop

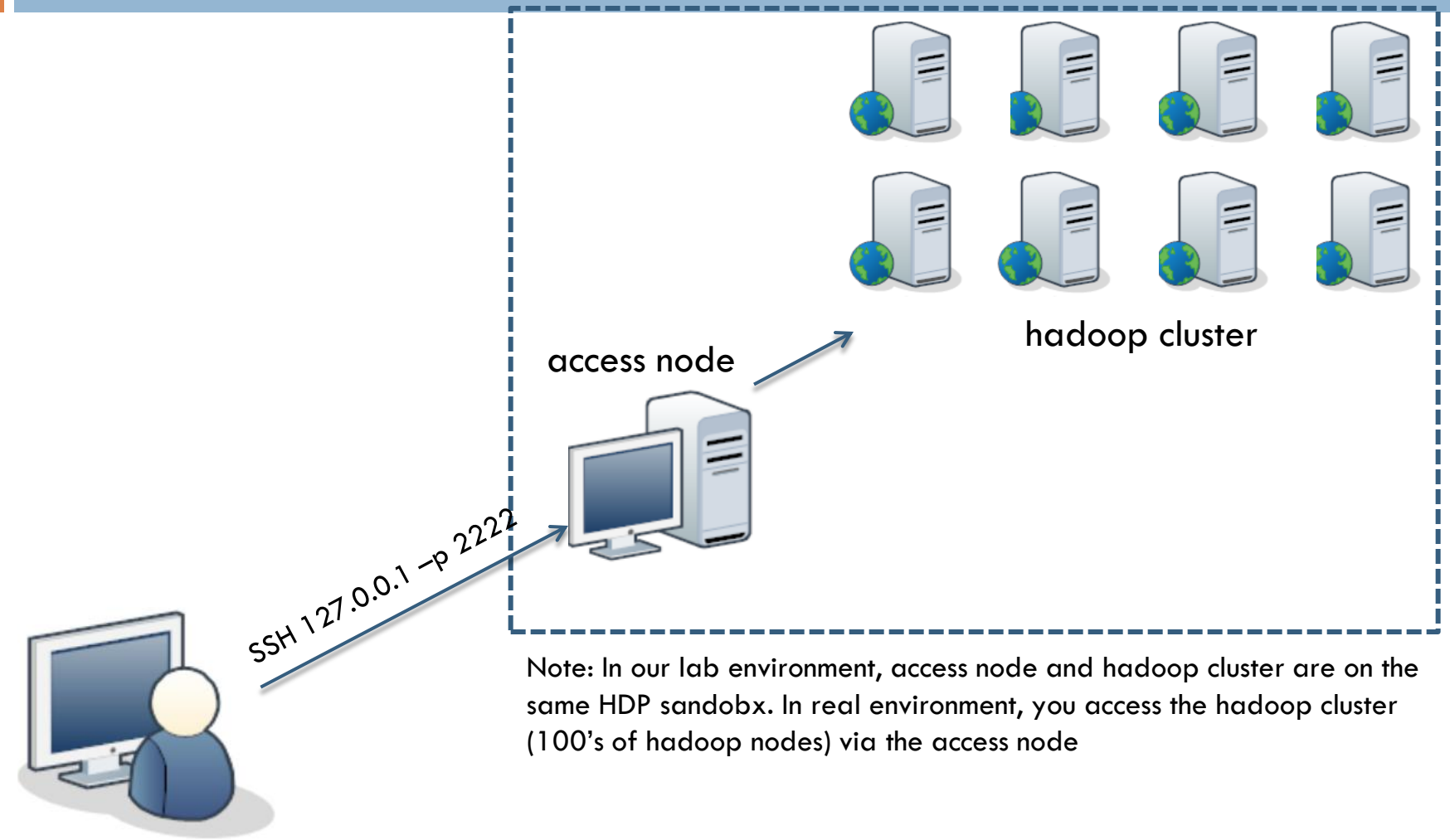
Sqoop Example

- Importing Data From MySQL Database

sqoop **import** -D mapred.job.queue.name=lowest --connect adbc:mysql://localhost/training --**username** datastudent --P --**table**=twitter.geotweets --**columns** "ID, TIMESTAMP, TWEETS" --fields-terminated-by '\t' --warehouse-dir /user/lab/tweets



Lab Environment



Note: In our lab environment, access node and hadoop cluster are on the same HDP sandobx. In real environment, you access the hadoop cluster (100's of hadoop nodes) via the access node

Lab 3 – Hadoop and WordCount

- Today's Lab
  - Hadoop Shell
  - WordCount – Hive
- Supplementary
  - WordCount – Java M/R
  - WordCount – Python Streaming

Lab 3 – Before We Start…

- Open Virtualbox
- Start HDP Sandbox
  - If you don't see HDP Sandbox, reload it
    - Please mark the computer ID so that I can tell the lab director!
- Start Putty and connect to Sandbox access node (client)

Lab 3 - Before We Start…

```
[root@sandbox ~]# ll
total 32
-rw-------. 1 root    root    2143 Dec 16 18:13 anaconda-ks.cfg
-rw-r--r--. 1 root    root    9436 Dec 16 18:13 install.log
-rw-r--r--. 1 root    root    3314 Dec 16 18:12 install.log.syslog
drwxr-xr-x  8 root    root    4096 Dec 16 19:33 ranger_tutorial
lrwxrwxrwx  1 root    root      48 Dec 16 19:15 start_ambari.sh -> /usr/lib/hue/tools/start_scripts/start_ambari.sh
lrwxrwxrwx  1 root    root      47 Dec 16 19:17 start_hbase.sh -> /usr/lib/hue/tools/start_scripts/start_hbase.sh
-rwxrwxrwx  1 vagrant vagrant  241 Dec 16 19:15 start_solr.sh
-rwxrwxrwx  1 vagrant vagrant   63 Dec 16 19:17 stop_solr.sh
[root@sandbox ~]# cd /home/lab
[root@sandbox lab]# ll
total 115364
drwxr-xr-x 5 nagios games     4096 Oct 12  2010 GeoText.2010-10-12
-rw-r--r-- 1 root   root  60973289 Jan 16 21:47 GeoText.2010-10-12.tgz
-rw-r--r-- 1 root   root  57139942 Jan 16 15:34 full_text.txt
-rwxrwxr-- 1 root   root      1027 Jan 23 21:38 sc_reducer.py
-rwxrwxr-- 1 root   root       537 Jan 23 21:38 wc_mapper.py
[root@sandbox lab]# cd GeoText.2010-10-12
[root@sandbox GeoText.2010-10-12]# ll
total 55820
-rw-r--r-- 1 nagios games     2695 Oct 12  2010 README.txt
-rw-r--r-- 1 nagios games 57139942 Oct 12  2010 full_text.txt
drwxr-xr-x 2 nagios games     4096 Jan 16 21:52 geo_eval
drwxr-xr-x 2 nagios games     4096 Jan 16 21:52 preproc
drwxr-xr-x 2 nagios games     4096 Jan 16 21:52 processed_data
[root@sandbox GeoText.2010-10-12]# cat full_text.txt | head
USER_79321756   2010-03-03T04:15:26     ÜT: 47.528139,-122.197916       47.528139       -122.197916     RT @USER_2ff4faca: IF SHE DO IT 1 MORE TIME.....IMA KNOC
&gt;haha. #cutthatout
USER_79321756   2010-03-03T04:55:32     ÜT: 47.528139,-122.197916       47.528139       -122.197916     @USER_77a4822d @USER_2ff4faca okay:) lol. Saying ok to bo
USER_79321756   2010-03-03T05:13:34     ÜT: 47.528139,-122.197916       47.528139       -122.197916     RT @USER_5d4d777a: YOURE A FAG FOR GETTING IN THE MIDDLE
OU ? A FUCKING NOBODY !!!!&gt;&gt;Lol! Dayum! Aye!
USER_79321756   2010-03-03T05:28:02     ÜT: 47.528139,-122.197916       47.528139       -122.197916     @USER_77a4822d yea ok..well answer that cheap as Sweden p
USER_79321756   2010-03-03T05:56:13     ÜT: 47.528139,-122.197916       47.528139       -122.197916     A sprite can disappear in her mouth - lil kim hmmmmm the
USER_79321756   2010-03-03T16:52:44     ÜT: 47.528139,-122.197916       47.528139       -122.197916     Lmao! I still get txt when AJ tweets before they even pos
s me dyin! @USER_a5b463b2 what's ur issue!
USER_79321756   2010-03-03T16:57:24     ÜT: 47.528139,-122.197916       47.528139       -122.197916     Alright twitters tryna take me over!
USER_79321756   2010-03-03T20:20:40     ÜT: 47.528139,-122.197916       47.528139       -122.197916     Just got to work. Got my pizza bagel and my raspberry ice
 not til 2. I just wanna get it done!:D
USER_79321756   2010-03-03T23:23:33     ÜT: 47.528139,-122.197916       47.528139       -122.197916     Just got a txt from my cousin! Yes! So happy for you @USE
USER_79321756   2010-03-03T23:37:36     ÜT: 47.528139,-122.197916       47.528139       -122.197916     Why is this woman in the bathroom everytime I'm in the ba
.
[root@sandbox GeoText.2010-10-12]#
```

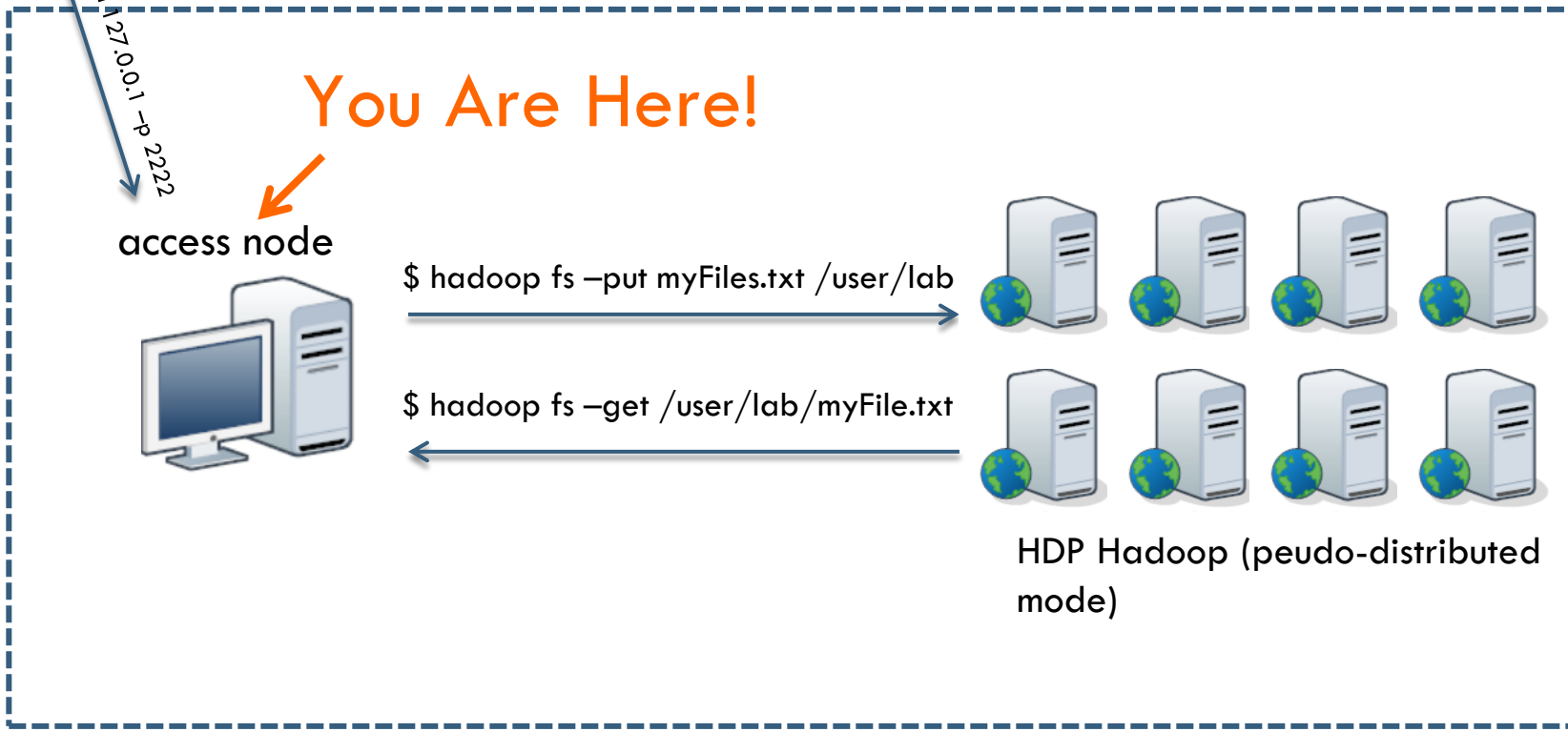
Twitter data directory

Make sure you can use cat to view the content of the data

Lab 3 – Now Let's Start…

Lab computer
Your PC/Mac

NOTE: Your client linux file system and hadoop filesystem (HDFS) are separate environment. First, you need to learn how to move files between the two file systems



Hadoop FileSystem Shell

- **hadoop fs –mkdir**
- **hadoop fs –ls**
- **hadoop fs –put**
- **hadoop fs –cat**
- **hadoop fs –get**
- **hadoop fs –rm**
- hadoop fs –rmdir
- hadoop fs –count
- hadoop fs –cp
- hadoop fs –du
- hadoop fs –mv
- hadoop fs –tail
- hadoop fs –getmerge

Lab 3 – Getting Files into HDFS

```
[root@sandbox GeoText.2010-10-12]# hadoop fs -ls /user/
Found 10 items
drwxr-xr-x   - hue       hdfs          0 2015-01-15 06:14 /user/4sq
drwxrwx---   - ambari-qa hdfs          0 2014-12-16 19:04 /user/ambari-qa
drwxr-xr-x   - guest     guest         0 2014-12-16 19:28 /user/guest
drwxr-xr-x   - hcat      hdfs          0 2014-12-16 19:13 /user/hcat
drwx------   - hive      hdfs          0 2014-12-16 19:08 /user/hive
drwxr-xr-x   - hue       hue           0 2014-12-16 19:27 /user/hue
drwxrwxr-x   - oozie     hdfs          0 2014-12-16 19:10 /user/oozie
drwx------   - root      hdfs          0 2015-01-24 05:19 /user/root
drwxr-xr-x   - solr      hdfs          0 2014-12-16 19:24 /user/solr
drwxr-xr-x   - hue       hdfs          0 2015-01-23 21:58 /user/twitter
[root@sandbox GeoText.2010-10-12]# hadoop fs -mkdir /user/lab/
[root@sandbox GeoText.2010-10-12]# hadoop fs -ls /user
Found 11 items
drwxr-xr-x   - hue       hdfs          0 2015-01-15 06:14 /user/4sq
drwxrwx---   - ambari-qa hdfs          0 2014-12-16 19:04 /user/ambari-qa
drwxr-xr-x   - guest     guest         0 2014-12-16 19:28 /user/guest
drwxr-xr-x   - hcat      hdfs          0 2014-12-16 19:13 /user/hcat
drwx------   - hive      hdfs          0 2014-12-16 19:08 /user/hive
drwxr-xr-x   - hue       hue           0 2014-12-16 19:27 /user/hue
drwxr-xr-x   - root      hdfs          0 2015-01-24 05:20 /user/lab
drwxrwxr-x   - oozie     hdfs          0 2014-12-16 19:10 /user/oozie
drwx------   - root      hdfs          0 2015-01-24 05:19 /user/root
drwxr-xr-x   - solr      hdfs          0 2014-12-16 19:24 /user/solr
drwxr-xr-x   - hue       hdfs          0 2015-01-23 21:58 /user/twitter
[root@sandbox GeoText.2010-10-12]# hadoop fs -put full_text.txt /user/lab
[root@sandbox GeoText.2010-10-12]# hadoop fs -ls /user/lab
Found 1 items
-rw-r--r--   1 root hdfs   57139942 2015-01-24 05:21 /user/lab/full_text.txt
[root@sandbox GeoText.2010-10-12]# hadoop fs -cat /user/lab/full_text.txt | head -n 5
USER_79321756   2010-03-03T04:15:20      ÜT: 47.528139,-122.197916       47.528139       -122.197916     RT @USER_2ff4faca: IF SHE DO IT 1 MORE TI
&gt;haha. #cutthatout
USER_79321756   2010-03-03T04:55:32      ÜT: 47.528139,-122.197916       47.528139       -122.197916     @USER_77a4822d @USER_2ff4faca okay:) lol.
USER_79321756   2010-03-03T05:13:34      ÜT: 47.528139,-122.197916       47.528139       -122.197916     RT @USER_5d4d777a: YOURE A FAG FOR GETTIN
OU ? A FUCKING NOBODY !!!!&gt;&gt;Lol! Dayum! Aye!
USER_79321756   2010-03-03T05:28:02      ÜT: 47.528139,-122.197916       47.528139       -122.197916     @USER_77a4822d yea ok..well answer that c
USER_79321756   2010-03-03T05:56:13      ÜT: 47.528139,-122.197916       47.528139       -122.197916     A sprite can disappear in her mouth - lil
cat: Unable to write to output stream.
[root@sandbox GeoText.2010-10-12]# hadoop fs -get /user/lab/full_text.txt full_text_2.txt
[root@sandbox GeoText.2010-10-12]# ll
total 111624
-rw-r--r-- 1 nagios games     2695 Oct 12  2010 README.txt
-rw-r--r-- 1 nagios games 57139942 Oct 12  2010 full_text.txt
-rw-r--r-- 1 root   root  57139942 Jan 24 05:23 full_text_2.txt
drwxr-xr-x 2 nagios games     4096 Jan 16 21:52 geo_eval
drwxr-xr-x 2 nagios games     4096 Jan 16 21:52 preproc
drwxr-xr-x 2 nagios games     4096 Jan 16 21:52 processed_data
[root@sandbox GeoText.2010-10-12]# hadoop fs -cp /user/lab/full_text.txt /user/lab/full_text_2.txt
[root@sandbox GeoText.2010-10-12]# hadoop fs -ls /user/lab
Found 2 items
-rw-r--r--   1 root hdfs   57139942 2015-01-24 05:21 /user/lab/full_text.txt
-rw-r--r--   1 root hdfs   57139942 2015-01-24 05:24 /user/lab/full_text_2.txt
[root@sandbox GeoText.2010-10-12]# hadoop fs -rm /user/lab/full_text_2.txt
15/01/24 05:25:20 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 360 minutes, Emptier interval = 0 minutes.
Moved: 'hdfs://sandbox.hortonworks.com:8020/user/lab/full_text_2.txt' to trash at: hdfs://sandbox.hortonworks.com:8020/user/root/.Trash/Current
[root@sandbox GeoText.2010-10-12]#
```

You don't see a /user/lab folder in HDFS yet

Create the lab folder in HDFS

Upload the full_text file to HDFS

Display the first few lines of HDFS file

Download a file from HDFS to access node

Make a copy of the HDFS file and rename

Delete a file in HDFS

Lab 3 – Hive WordCount

Open Hive Interactive CLI

```
[root@sandbox GeoText.2010-10-12]# hive
15/01/24 05:36:27 WARN conf.HiveConf: HiveConf of name hive.optimize.mapjoin.mapreduce does not exist
15/01/24 05:36:27 WARN conf.HiveConf: HiveConf of name hive.heapsize does not exist
15/01/24 05:36:27 WARN conf.HiveConf: HiveConf of name hive.server2.enable.impersonation does not exist
15/01/24 05:36:27 WARN conf.HiveConf: HiveConf of name hive.auto.convert.sortmerge.join.noconditionaltask does not exist

Logging initialized using configuration in file:/etc/hive/conf/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/hive/lib/hive-jdbc-0.14.0.2.2.0.0-2041-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
hive>
```

Display the current tables

```
    > show tables;
OK
full_text
sample_07
sample_08
tweets1
Time taken: 3.234 seconds, Fetched: 4 row(s)
hive> create table full_text (line string);
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. AlreadyExistsException(message:Table full_text already exists)
hive> drop table full_text;
OK
Time taken: 0.794 seconds
```

Create an empty table called full_text

```
hive> create table full_text (line string);
OK
Time taken: 0.405 seconds
```

Load the full_text.txt file from HDFS into the full_text table in Hive

```
hive> load data inpath '/user/lab/full_text.txt' overwrite into table full_text;
Loading data to table default.full_text
Table default.full_text stats: [numFiles=1, numRows=0, totalSize=57139942, rawDataSize=0]
OK
Time taken: 1.48 seconds
```

Select to display the first 2 lines

```
hive> select * from full_text limit 2;
OK
USER_79321756   2010-03-03T04:15:26     ÜT: 47.528139,-122.197916       47.528139       -122.197916     RT @USER_2ff4faca: IF SHE DO IT 1 MORE TIME.....IMA KNOCK HER DAMN KOOFIE OF
&gt;haha. #cutthatout
USER_79321756   2010-03-03T04:55:32     ÜT: 47.528139,-122.197916       47.528139       -122.197916     @USER_77a4822d @USER_2ff4faca okay:) lol. Saying ok to both of yall about to
Time taken: 1.088 seconds, Fetched: 2 row(s)
```

Word Count query!

```
hive> create table wordcount as
    > select word, count(1) as count from
    > (select explode(split(line,'[\s+ +\t+]')) as word from full_text) w
    > group by word
    > order by count desc;
```

```
hive> create table wordcount as
    > select word, count(1) as count from
    > (select explode(split(line,'[\s+ +\t+]')) as word from full_text) w
    > group by word
    > order by count desc;
Query ID = root_20150124053838_1e737143-4823-4df5-8483-959c0e33e8bb
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1422074871964_0001, Tracking URL = http://sandbox.hortonworks.com:8088/proxy/application_1422074871964_0001/
Kill Command = /usr/hdp/2.2.0.0-2041/hadoop/bin/hadoop job  -kill job_1422074871964_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2015-01-24 05:38:56,317 Stage-1 map = 0%,  reduce = 0%
2015-01-24 05:39:13,601 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 15.37 sec
2015-01-24 05:39:14,650 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 16.65 sec
2015-01-24 05:39:26,471 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 24.05 sec
MapReduce Total cumulative CPU time: 24 seconds 50 msec
Ended Job = job_1422074871964_0001
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1422074871964_0002, Tracking URL = http://sandbox.hortonworks.com:8088/proxy/application_1422074871964_0002/
Kill Command = /usr/hdp/2.2.0.0-2041/hadoop/bin/hadoop job  -kill job_1422074871964_0002
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2015-01-24 05:39:35,374 Stage-2 map = 0%,  reduce = 0%
2015-01-24 05:39:44,985 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 5.22 sec
2015-01-24 05:39:56,195 Stage-2 map = 100%,  reduce = 80%, Cumulative CPU 12.43 sec
2015-01-24 05:39:58,370 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 14.27 sec
MapReduce Total cumulative CPU time: 14 seconds 270 msec
Ended Job = job_1422074871964_0002
Moving data to: hdfs://sandbox.hortonworks.com:8020/apps/hive/warehouse/wordcount
Table default.wordcount stats: [numFiles=1, numRows=865487, totalSize=13935272, rawDataSize=13069785]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 24.05 sec   HDFS Read: 57140175 HDFS Write: 27997246 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 14.27 sec   HDFS Read: 27997661 HDFS Write: 13935355 SUCCESS
Total MapReduce CPU Time Spent: 38 seconds 320 msec
OK
Time taken: 85.974 seconds
hive> select * from wordcount limit 5;
OK
        588285
ÜT:     339083
I       110427
a       81038
the     78480
```

The WordCount query

Launching a MapReduce job

Reporting progress

Done!

Select query to display the first 5 word count

Lab 3 – Supplementary

- To learn more about Hadoop shell commands, check out the documentations

http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-common/FileSystemShell.html

Lab 3 – Supplementary

- *Java M/R WordCount*

1. Download the Shakespeare file
   - Shakespeare dataset has been uploaded to Blackboard "datasets" page
2. Put the shakespeare file into HDFS
3. Find your mapreduce-example jar file
   - $ find /usr -name *mapreduce-example*
4. Run java M/R wordcount example
   - $ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar wordcount /user/shaohua/shakespeare /user/shaohua/shakespeare-wc-out

Change the path accordingly

- View results

Lab 3 – Supplementary

- *Python Streaming - WordCount*

1. Download the Shakespeare file
   - Shakespeare dataset has been uploaded to Blackboard "datasets" page
2. Put the shakespeare file into HDFS
3. Find your mapreduce-example jar file
   - $ find /usr -name *hadoop-streaming*
4. Run Hadoop streaming wordcount

   Change the path accordingly

   - $ hadoop jar /usr/local/hadoop-2.6.0/share/hadoop/mapreduce/hadoop-streaming-2.6.0.jar **-file** /Users/DSinmotion/Ryerson/Demos/scripts/wc_mapper.py **-mapper** /Users/DSinmotion/Ryerson/Demos/scripts/wc_mapper.py **-file** /Users/DSinmotion/Ryerson/Demos/scripts/wc_reducer.py **-reducer** /Users/DSinmotion/Ryerson/Demos/scripts/wc_reducer.py **-input** /user/shaohua/shakespeare **-output** /user/shaohua/shakespeare-wc-out-1
5. View results