Hadoop

Copyright 2015 by Ernst Henle

Hadoop Intro (1)

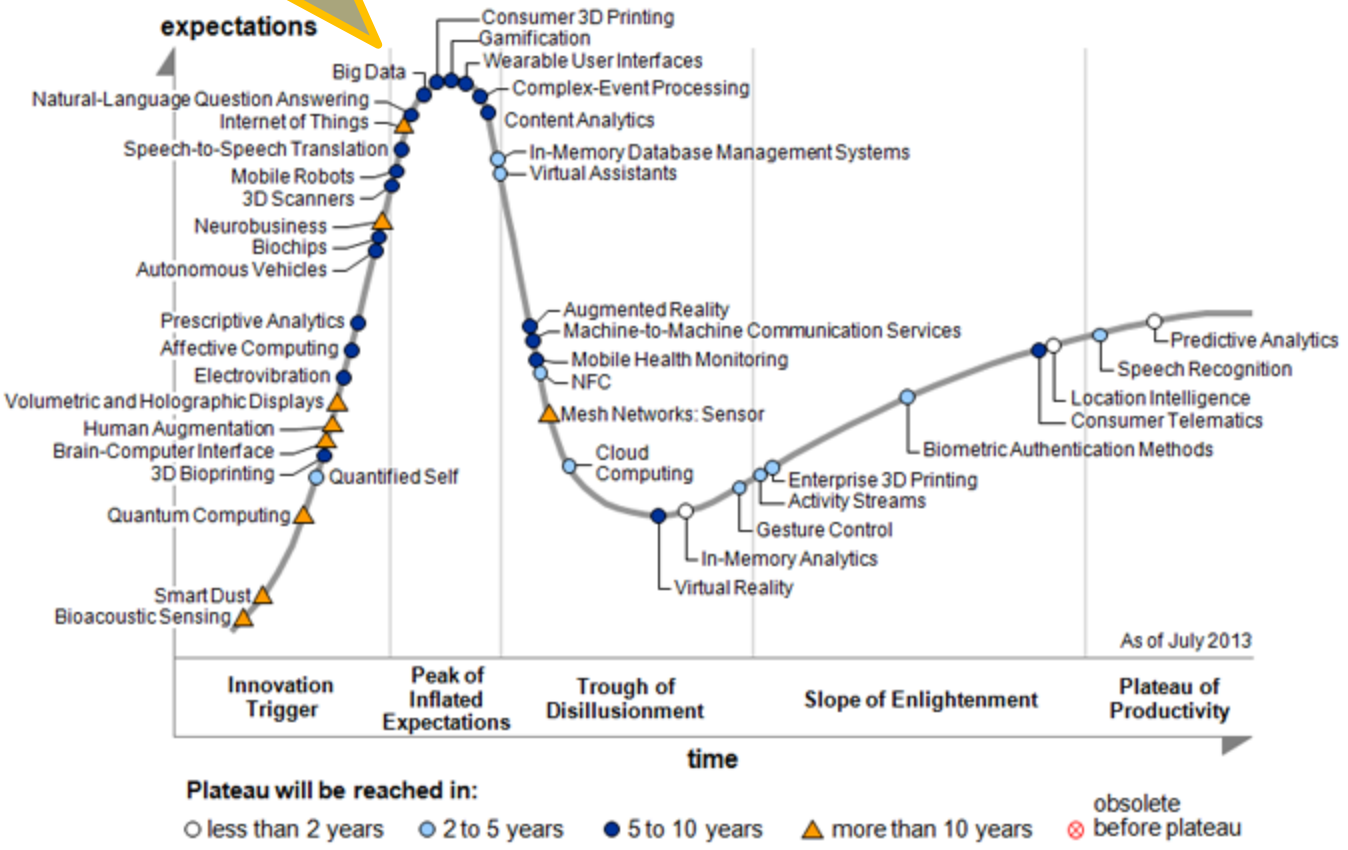
- Hadoop was created in 2008 at Yahoo by Doug Cutting and Mike Cafarella. Their work was based on two papers by Google:
  - **The Google File System**
  - **MapReduce: Simplied Data Processing on Large Clusters**

Hadoop Intro (2)

- Hadoop is
  - for big data
  - a framework for storing and processing large datasets (> 100 GB)
  - Apache open source software
  - inefficient and awkward for small datasets (< 10 GB)

Hadoop Intro (3)

Big Data on the BI Hype Curve



Hadoop Intro (4)

- Hadoop's most important two components
  1. HDFS for storage (Hadoop Distributed File System)
  2. MapReduce for processing (Supported programming pattern for Hadoop tasks)
- Other required Hadoop components
  1. YARN (Resource management of MapReduce tasks)
  2. Common (Basic Hadoop libraries)

Hadoop Intro (5)

- Commodity Hardware (typical: 2 processors, 4 GB)
- Scalable (Vary number of nodes)
- Distributed processing to increase processing power
- Data replication to make storage more secure
- Avoid Data roundtrips
- Write Once Read Many and Update never! (http://en.wikipedia.org/wiki/Write_once_read_many)
- Streaming Access Pattern (Sequential Access of large chunks)
- Large block size (64 MB) enables fast sequential read. Data are streamed off the storage device while maintaining the maximum read rate. (Compare to common 4 KB block)

Hadoop Intro (6)

- Hadoop daemons:
  - Master Services:
    - NameNode: Manages HDFS
    - Secondary NameNode (Helps NameNode)
    - JobTracker: Manages TaskTrackers and MapReduce for a cluster
  - Slave Services:
    - DataNode: Executes MapReduce
    - TaskTracker: Manages MapReduce for a DataNode

Hadoop Intro (7)

- Ecosystem
  - Hive: Data warehouse on Hadoop
  - HiveQL SQL-like language for Hive
  - Impala: SQL-like query engine (Cloudera)
  - Sqoop: SQL to Hadoop. Transfers data between Hadoop and a relation database.
  - Oozie: Workflow scheduler for Hadoop. Scheduling is arranged like a DFD with DAG constraints
  - Flume: Collects and aggregates log files in Hadoop
  - Pig: Dataflow language and programming tool for MapReduce programs on Hadoop
  - Pig Latin: Language for Pig.
  - Mahout: Machine Learning algorithms that utilize the MapReduce pattern and run on Hadoop
  - Hue: Web-based interactive file browser

HDFS: How it works

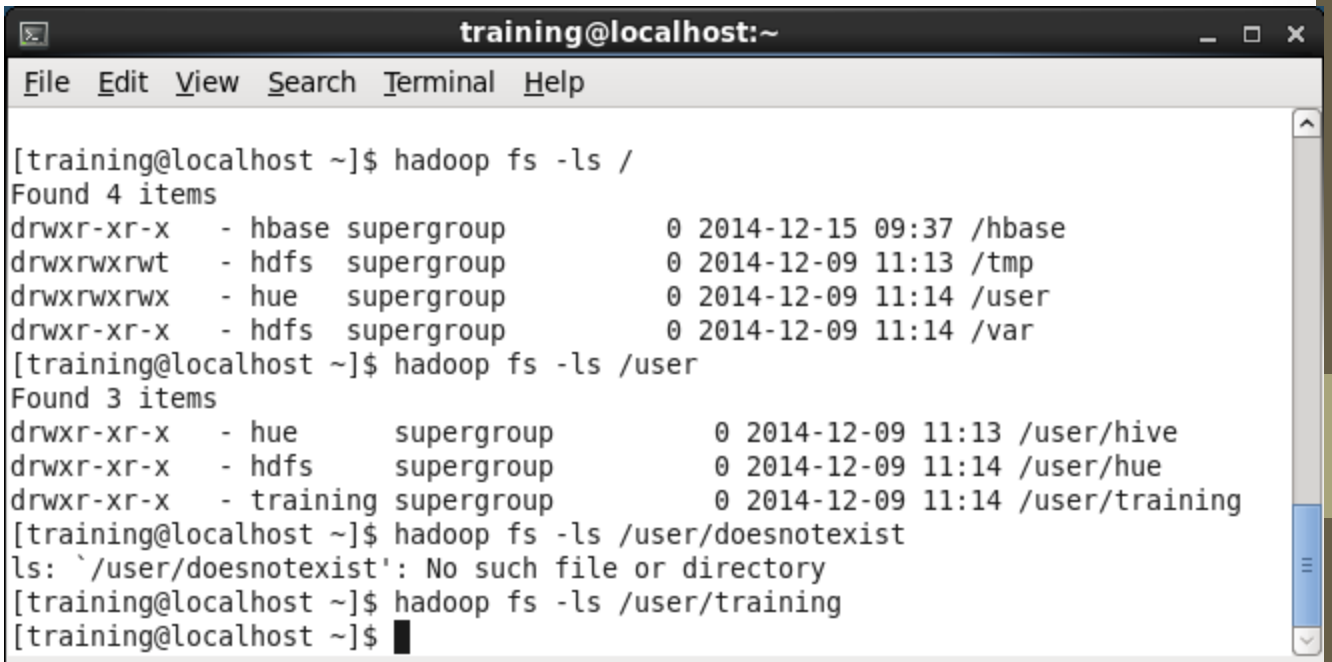
- See: HowHDFSworksByManeeshVarshney.pdf

HDFS Lab

"So you want to hire me as a Data Scientist for Intelligent Virtualized Deep Machine Learning Real-time Big Data in the Cloud for Social Networks? Ok, but if you also want Hadoop, increase my salary by 50%."

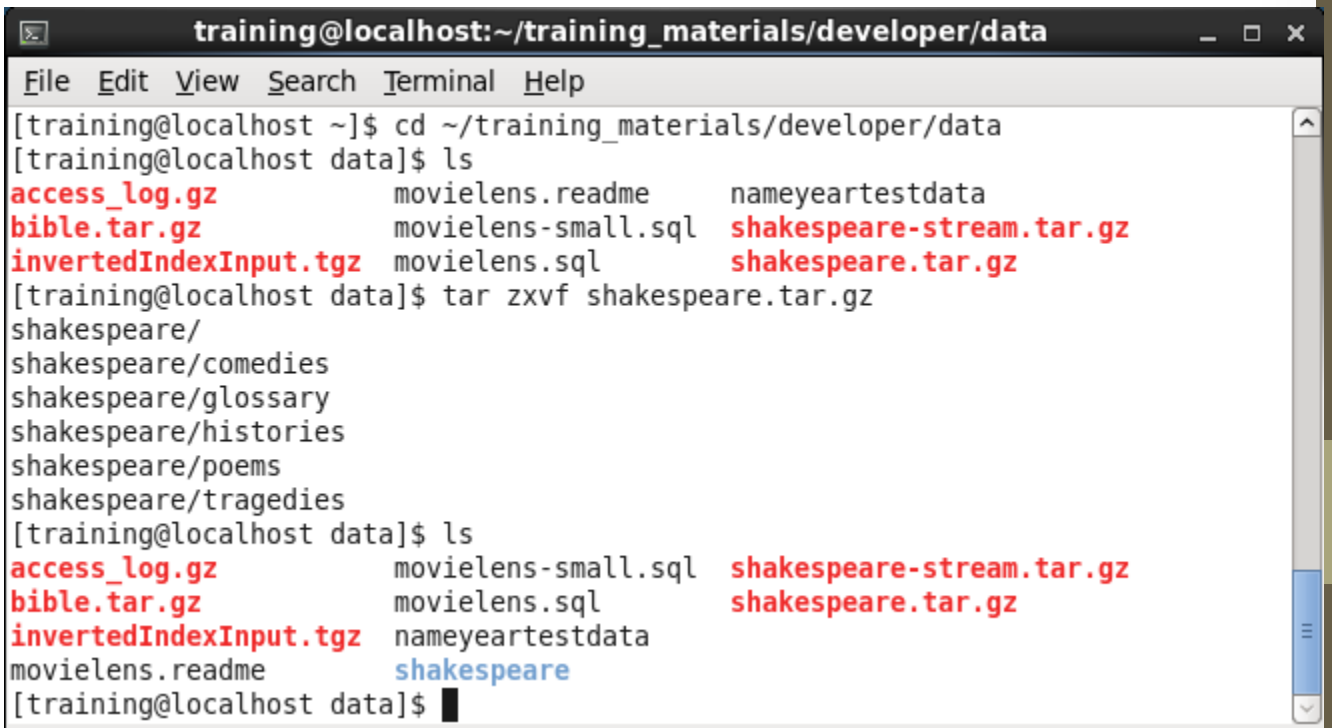
HDFS Lab (1)

- Enter these commands into the console to list some folders and files inside HDFS:
  - `$ hadoop fs -ls /`
  - `$ hadoop fs -ls /user`
  - `$ hadoop fs -ls /user/doesnotexist`
  - `$ hadoop fs -ls /user/training`
- "fs" stands for file system (HDFS). "ls" is an argument to list HDFS files.



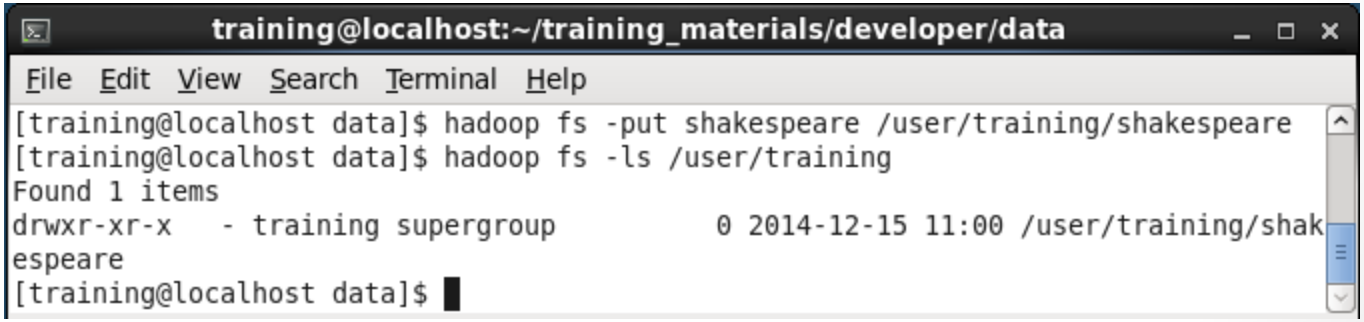
HDFS Lab (2)

- Enter these commands into the console to find and expand the shakespeare tar on the Linux OS :
  - `$ cd ~/training_materials/developer/data`
  - `$ ls`
  - `$ tar zxvf shakespeare.tar.gz`
  - `$ ls`



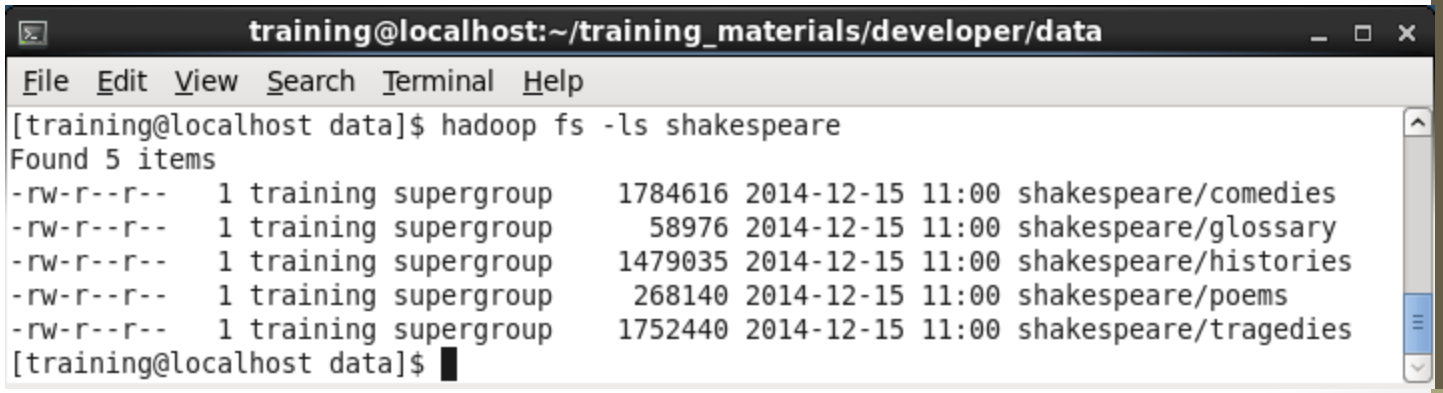
HDFS Lab (3)

- Enter these commands into the console to place the shakespeare directory into HDFS under training and then verify that the directory is in HDFS:
  - `$ hadoop fs -put shakespeare /user/training/shakespeare`
  - `$ hadoop fs -ls /user/training`



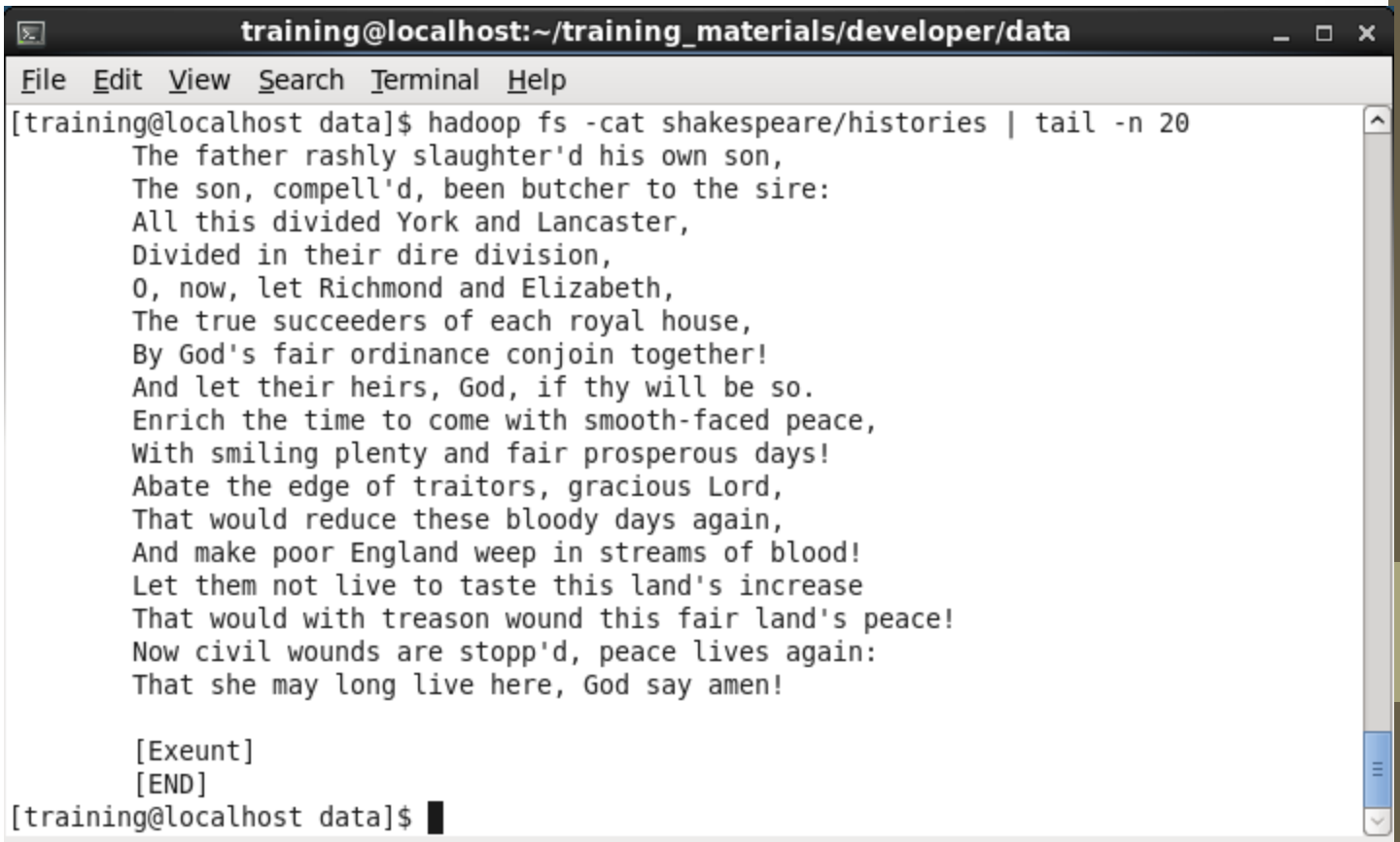
HDFS Lab (4)

- Enter this command into the console to list the 5 files in the shakespeare directory of HDFS:
  - `$ hadoop fs -ls /user/training/shakespeare`



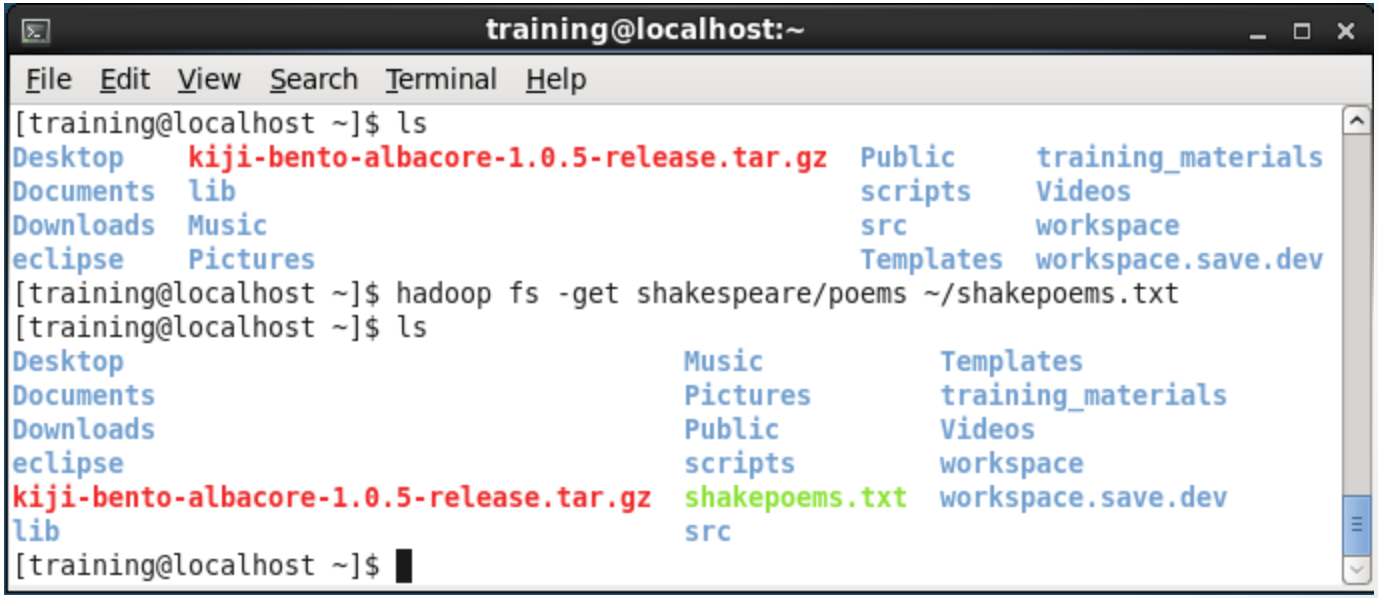
HDFS Lab (5)

- Enter this command into the console to read the last 20 lines of the histories file in HDFS:
  - `$ hadoop fs -cat shakespeare/histories | tail -n 20`



HDFS Lab (6)

- Enter the following commands to retrieve files from HDFS and list them in the Linux OS:
  - `$ ls`
  - `$ hadoop fs -get shakespeare/poems ~/shakepoems.txt`
  - `$ ls`



Sqoop Lab

- **Sq**l + Had**oop** → **Sqoop**
- Use Sqoop to share between a RDBMS and Hadoop

Sqoop Lab (1)

- Use this Sqoop command to familiarize yourself with Sqoop:
  - $ **sqoop help**

- Use these sqoop commands to list mysql databases on localhost and then tables in one of those databases:
  - $ **sqoop list-databases --connect jdbc:mysql://localhost --username training --password training**
  - $ **sqoop list-tables --connect jdbc:mysql://localhost/movielens --username training --password training**

Sqoop Lab (2)

- Use this Sqoop command to import data from a mysql table to an HDFS table. The fields of a table record (row) will be separated by tabs in the corresponding record of the HDFS file:
  - $ **sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movie**

- Note that Sqoop constructs some SQL statements for MySQL and that it converts the SQL statement into a MapReduce job that only has a Map component:
  - **Test SQL: SELECT t.* FROM `movie` AS t LIMIT 1**
  - **SELECT MIN(`id`), MAX(`id`) FROM `movie`**

- Use this HDFS command to list the newly imported file(s) and then view the last part of one of the files:
  - $ **hadoop fs -ls movie**
  - $ **hadoop fs -tail movie/part-m-00000**

To run this command a second time you will need to avoid a naming collision:
sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movie --target-dir /movie2

Sqoop Lab (3)

- Use this Sqoop command to import data from a mysql table to an HDFS table. The fields of a table record (row) will be separated by tabs in the corresponding record of the HDFS file:
  - $ **`sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movierating`**

- Use these HDFS commands to list the newly imported file(s) and then view the last part of one of the files:
  - $ **`hadoop fs -ls movierating`**
  - $ **`hadoop fs -tail movierating/part-m-00000`**

To run this command a second time you will need to avoid a naming collision:
sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movierating --target-dir /movierating2

Hive Lab

"It does look similar—but this one is powered by Hadoop"

Hive Lab (1)

- Check that we have movie and movierating in HDFS
  - $ **`hadoop fs -cat movie/part-m-00000 | head`**
  - $ **`hadoop fs -cat movierating/part-m-00000 | head`**

- Start Hive
  - $ **`hive`**

- Tell Hive that the HDFS files, movie and movierating, are tables:
  - **hive> CREATE EXTERNAL TABLE movie (id INT, name STRING, year INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LOCATION '/user/training/movie';**
  - **hive> CREATE EXTERNAL TABLE movierating (userid INT, movieid INT, rating INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LOCATION '/user/training/movierating';**

To run this command a second time you will need to avoid a naming collision:
hive> DROP TABLE **`movie`**
hive> DROP TABLE **`movie`**rating

Hive Lab (2)

- Ask Hive to show tables and provide metadata on these tables
  - `hive> SHOW TABLES;`
  - `hive> DESCRIBE movie;`
  - `hive> DESCRIBE movierating;`

- Use Hive to find information from your tables
  - **hive> SELECT * FROM movie WHERE year < 1925;**
  - **hive> SELECT * FROM movie WHERE year < 1925 AND year != 0 ORDER BY name;**
  - **hive> SELECT * FROM movierating WHERE userid=149;**

Hive Lab (3)

- hive> CREATE TABLE USERRATING (userid INT, numratings INT, avgrating FLOAT);
- hive> insert overwrite table userrating SELECT userid,COUNT(userid),AVG(rating) FROM movierating GROUP BY userid;

- hive> SELECT AVG(rating) FROM movierating WHERE userid=149;
- hive> SELECT userid, COUNT(userid),AVG(rating) FROM movierating where userid < 10 GROUP BY userid;

Hive Lab (4)

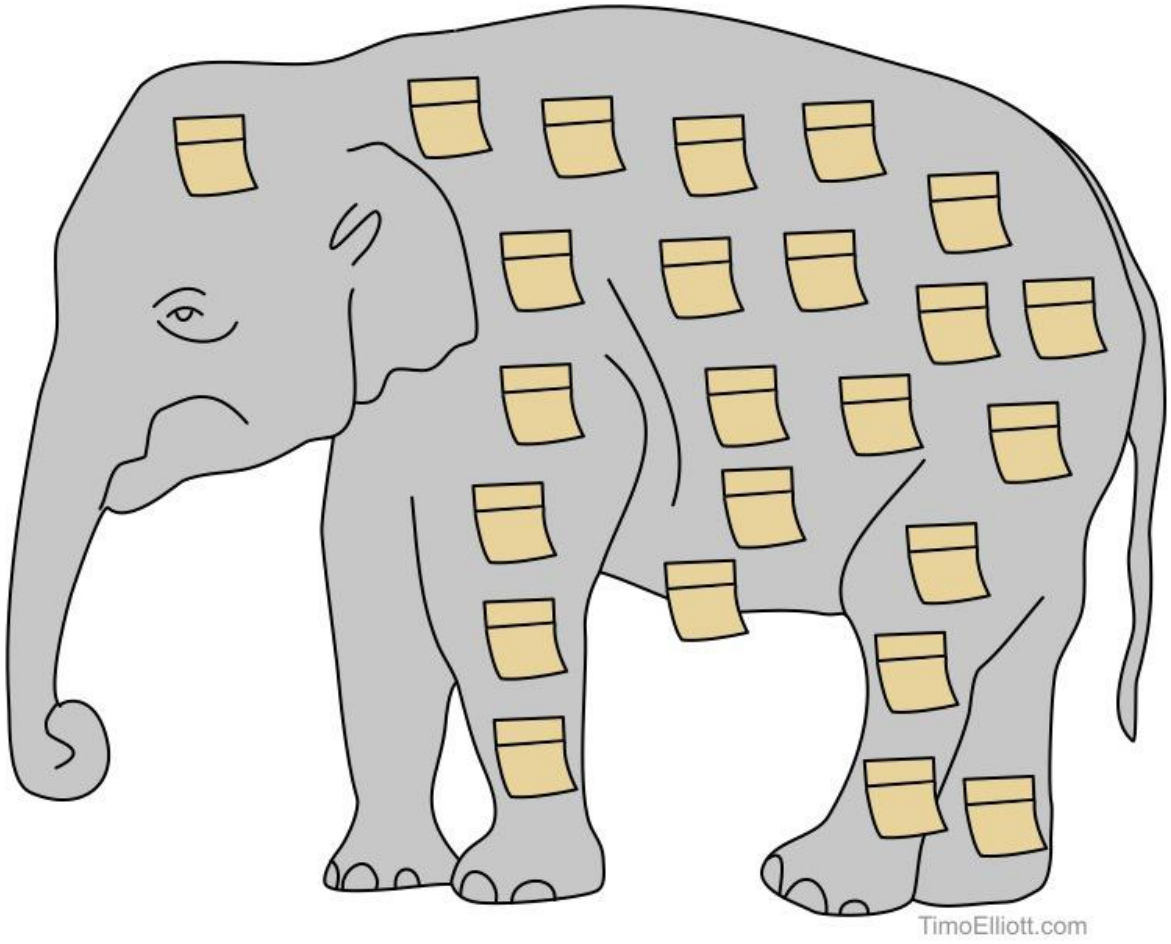
- Query with Join
  - hive> select movieid,rating,name from movierating join movie on movierating.movieid=movie.id where userid=149;
- hive> QUIT;

- Take-home Exercise
  - What is oldest movie in the database that has a top rating?

Hue and Impala

- Use Impala to execute “HiveQL”
- Open Firefox
- Start Hue by clicking on the Hue link in the favorites
- Select Impala and Query Tab
- Enter Query into Query text box for Impala:
  - **select movieid,rating,name from movierating join movie on movierating.movieid=movie.id where userid=149;**

Break



An early Hadoop prototype...

MapReduce (0)

MapReduce (1)

- MapReduce is a pattern (programming model) designed to work in Hadoop
- MapReduce works with HDFS
- The two primary ideas behind MapReduce
  - Send the program to the data as opposed to the data to the program
  - Make use of distributed data where each chunk of data has its own CPUs
- When a MapReduce job is started, then Hadoop sends Map and Reduce jobs to the data nodes. Hadoop manages all the details of data passing among data nodes.
  - The Map portion of the computation occurs on the individual data nodes where the data reside.
  - The reduce portion of the computation occurs on some of the data nodes (not necessarily where the data reside). In some cases part of the reduce operation can occur on the data node where the data reside.

MapReduce (2)

- MapReduce uses three stages:
  - Map
  - Shuffle
  - Reduce
- Then, why don't we call MapReduce: MapShuffleReduce?
  - Because: In most cases, the programmer need only be concerned with Map and Reduce. The Shuffle and Sorting is taken care of by Hadoop.
  - To achieve scalable programs, Hadoop takes care off:
    - Creating tasks on the various data nodes
    - Tracking Tasks
    - Moving data among data nodes
    - File I/O, networking, process synchronization, recovery from failures, re-running jobs, splitting input, Moving Key-Value-Pairs from Map to Reduce
    - Outputs results

MapReduce (3)

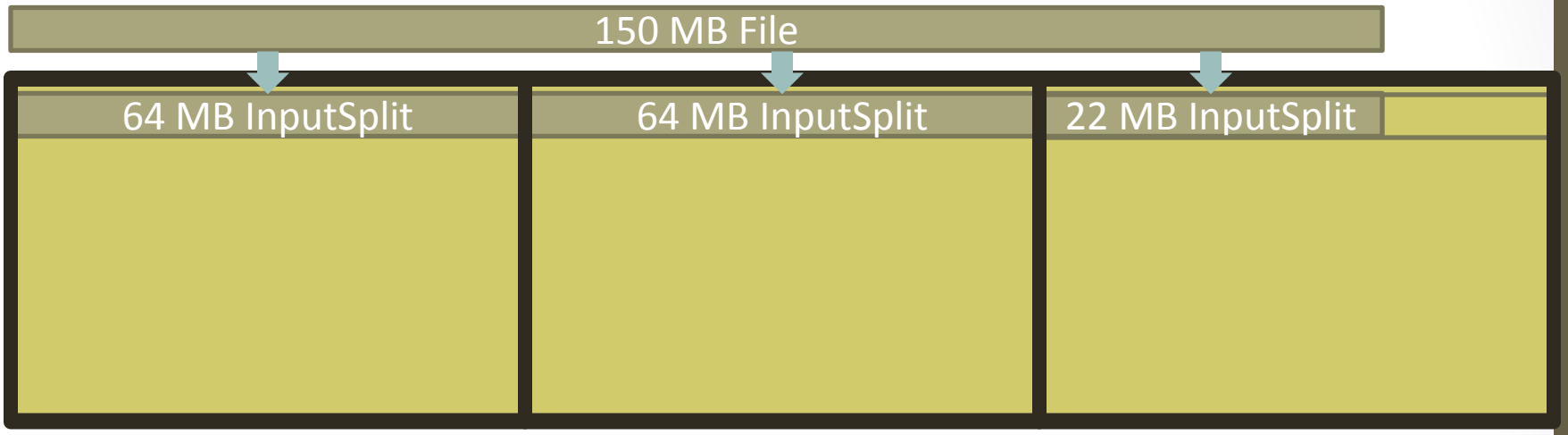
- Parallelization Rules in MapReduce
  - Map tasks occur in parallel independent of each other
  - Reducers can start before Mappers are done. But, reducers cannot complete before Mapping and Shuffle & Sort have completed.
  - Reduce tasks occur in parallel independent of each other

MapReduce (4)

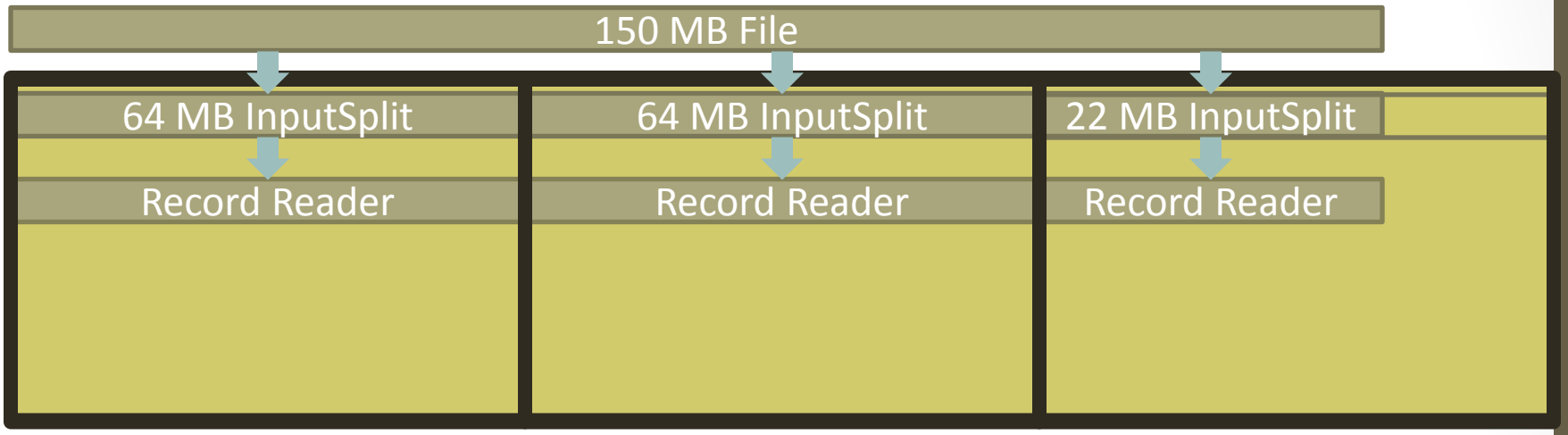
MapReduce (5)

150 MB File
64 MB InputSplit
64 MB InputSplit
22 MB InputSplit

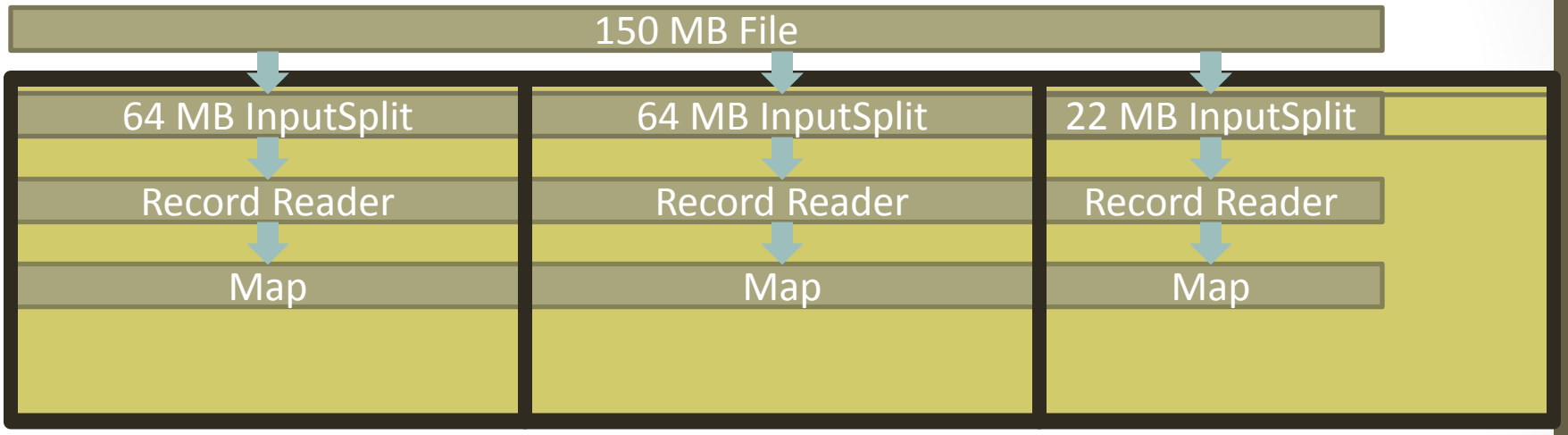
MapReduce (6)


150 MB File
64 MB InputSplit
64 MB InputSplit
22 MB InputSplit

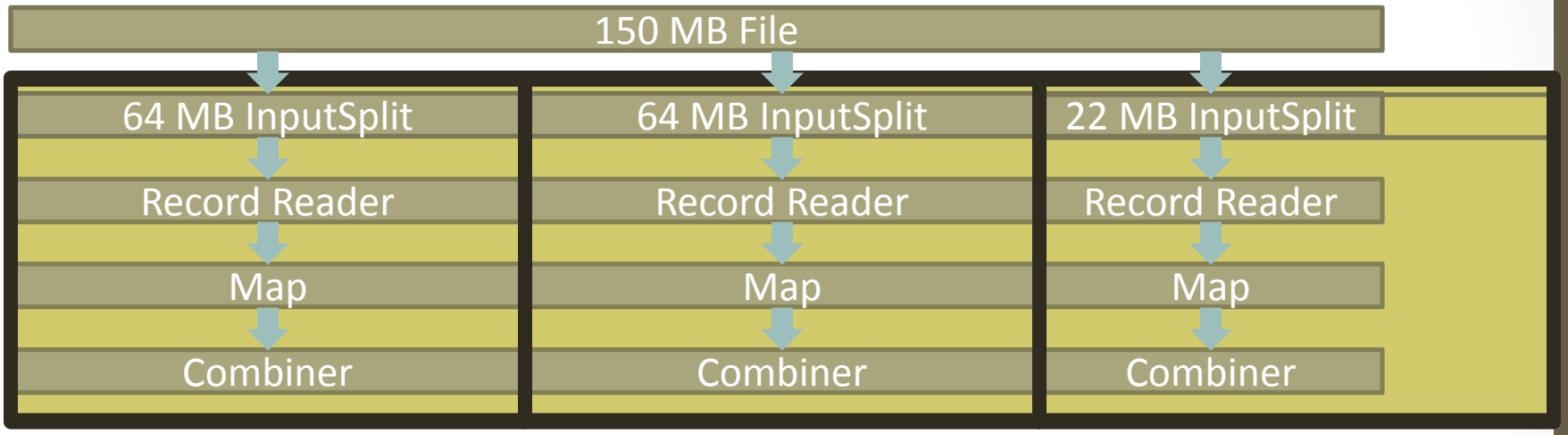
MapReduce (7)


150 MB File
64 MB InputSplit
64 MB InputSplit
22 MB InputSplit
Record Reader
Record Reader
Record Reader

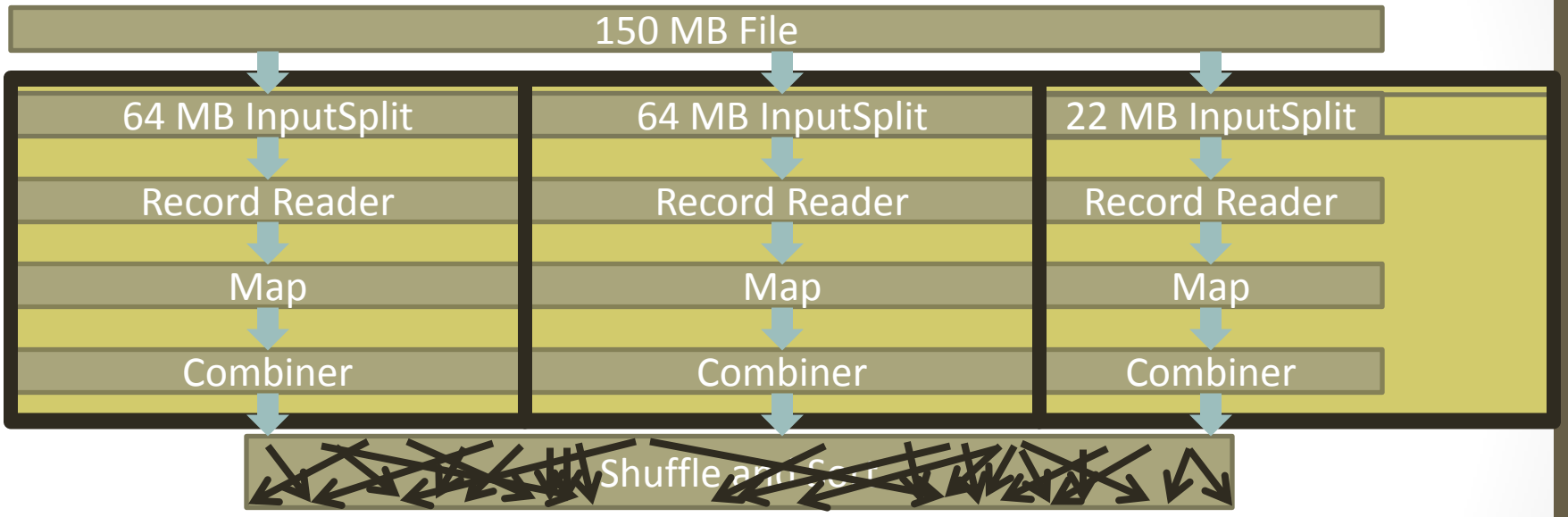
MapReduce (8)


150 MB File
64 MB InputSplit
Record Reader
Map
64 MB InputSplit
Record Reader
Map
22 MB InputSplit
Record Reader
Map

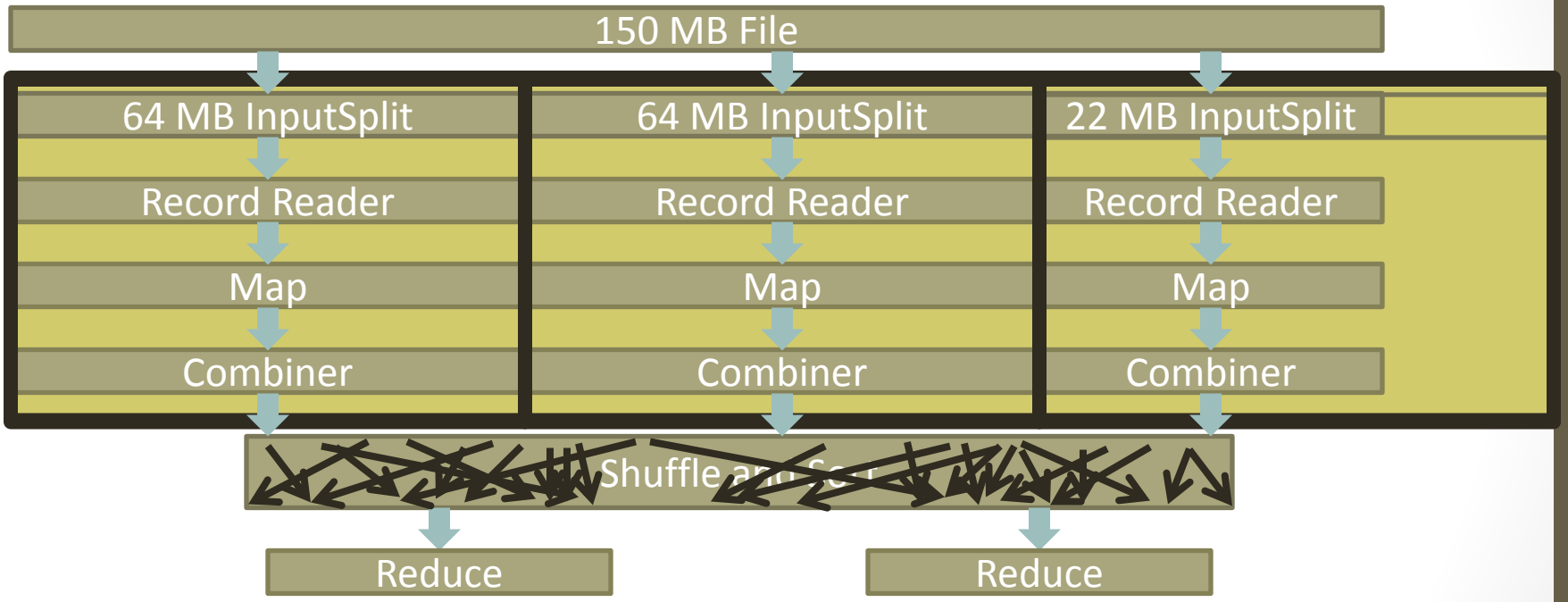
MapReduce (9)


150 MB File
64 MB InputSplit
Record Reader
Map
Combiner
64 MB InputSplit
Record Reader
Map
Combiner
22 MB InputSplit
Record Reader
Map
Combiner

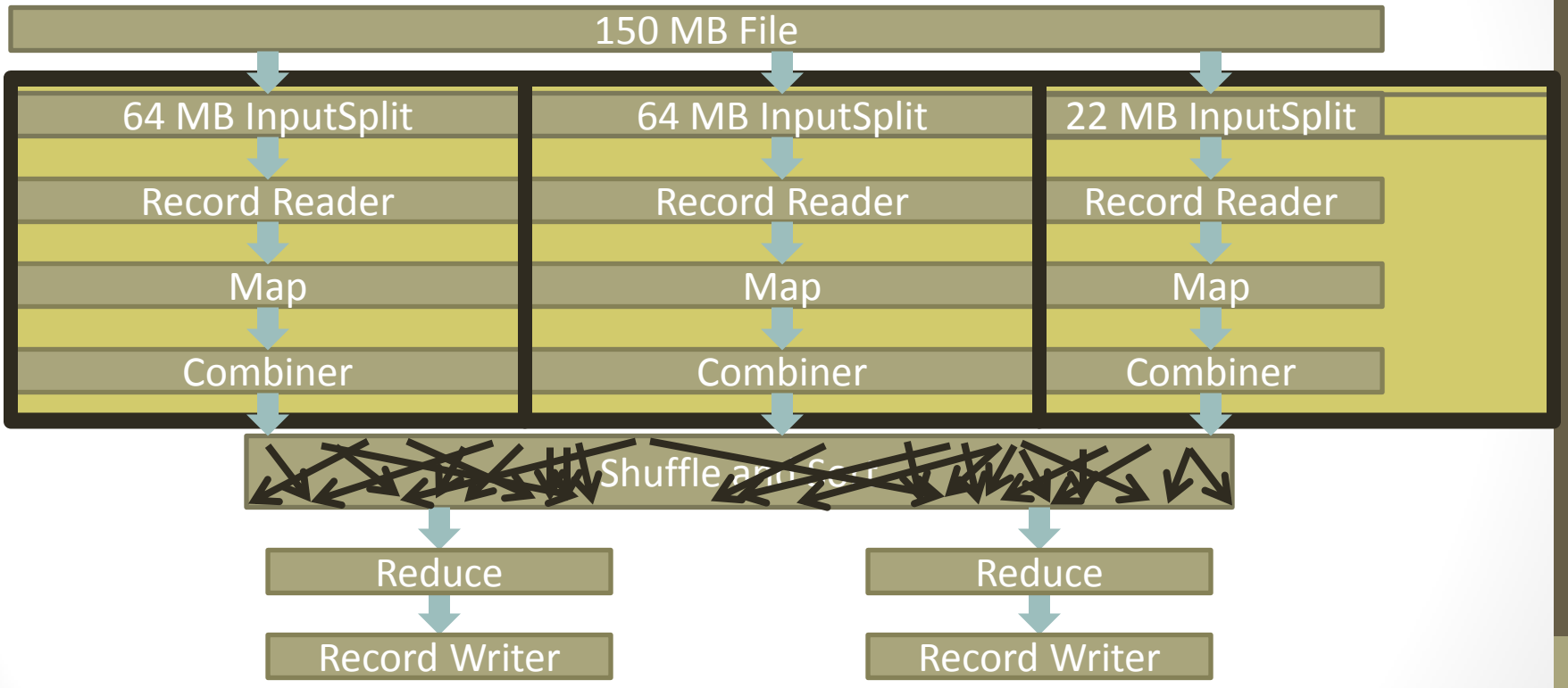
MapReduce (10)



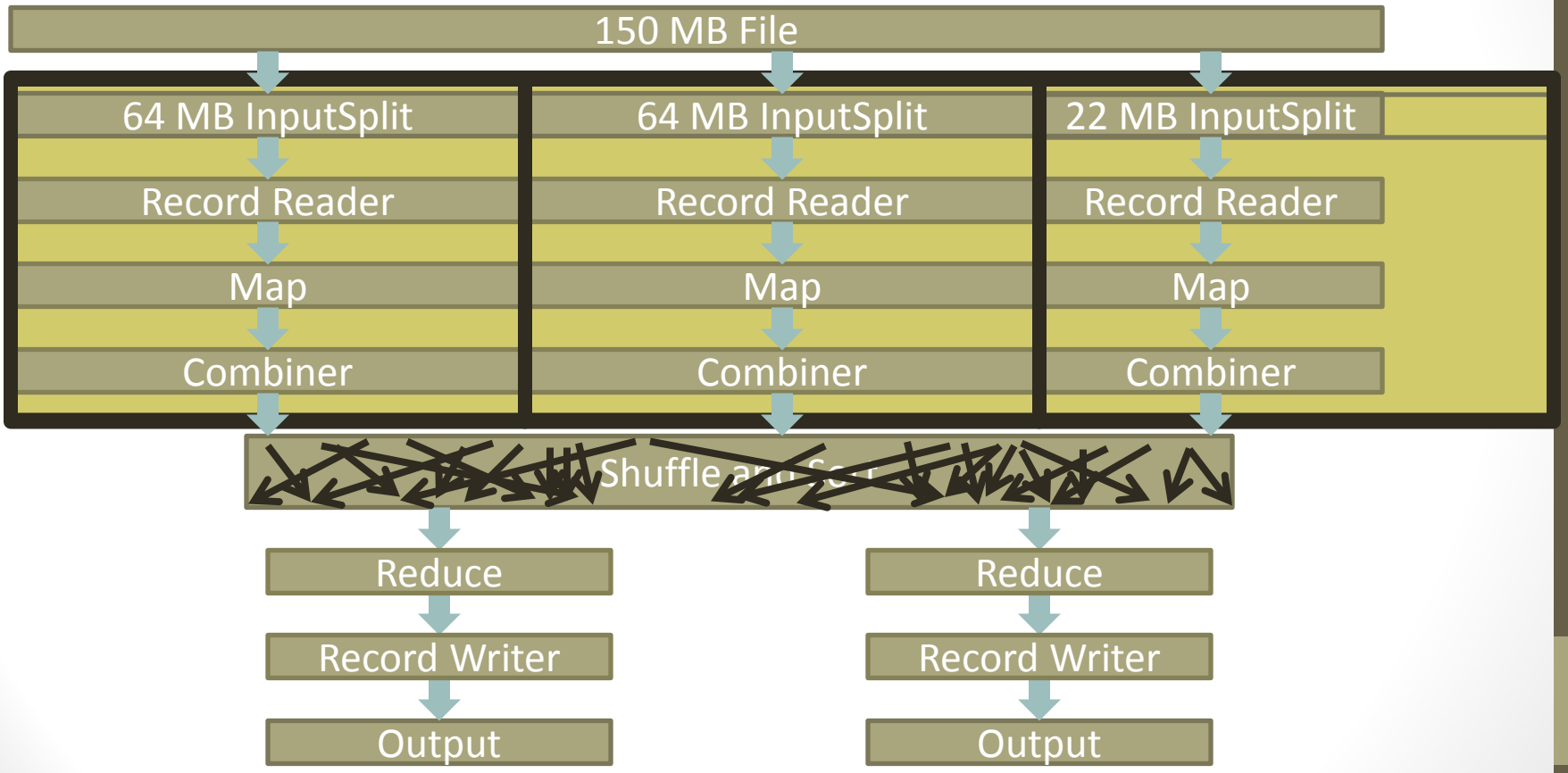
MapReduce (11)



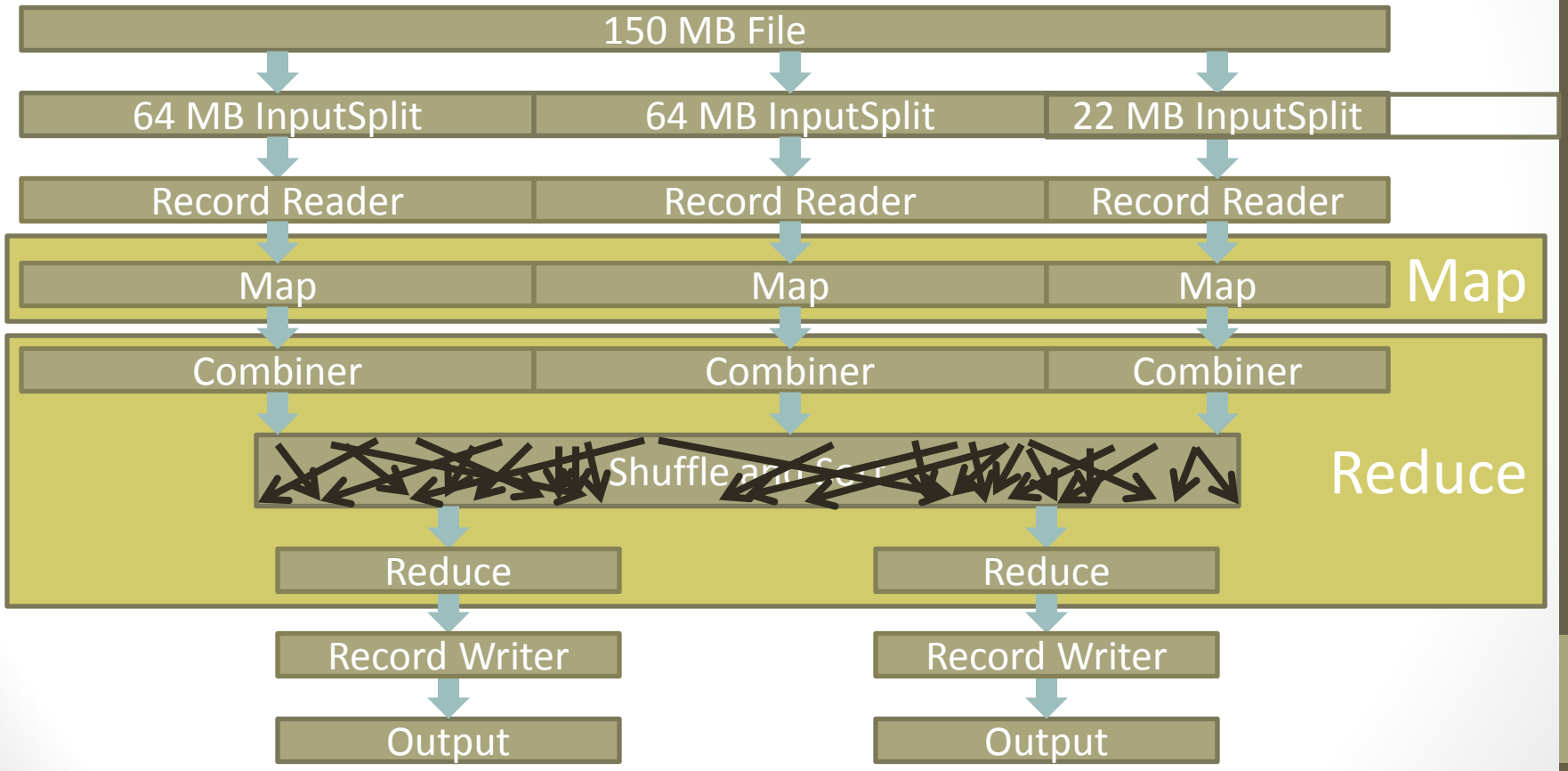
MapReduce (12)



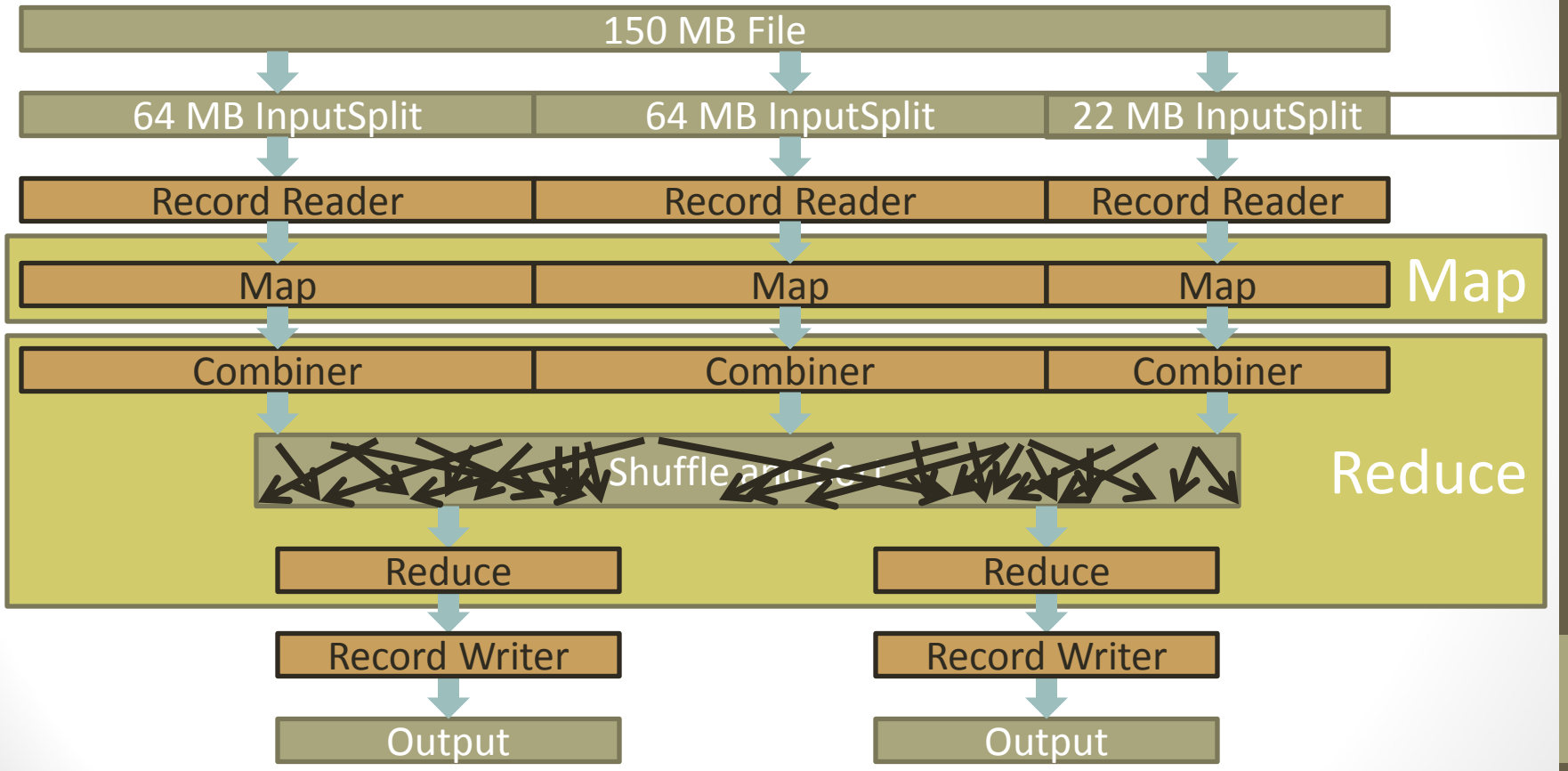
MapReduce (13)



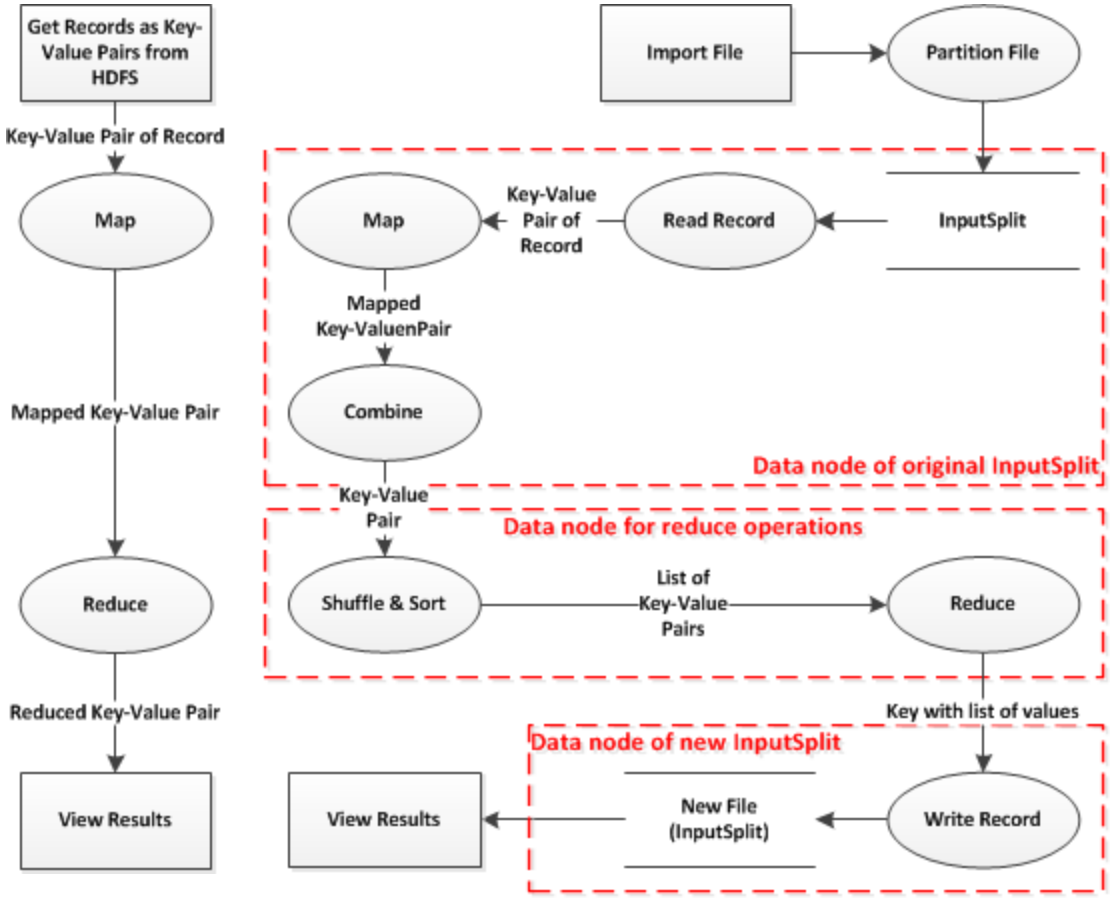
MapReduce(14)



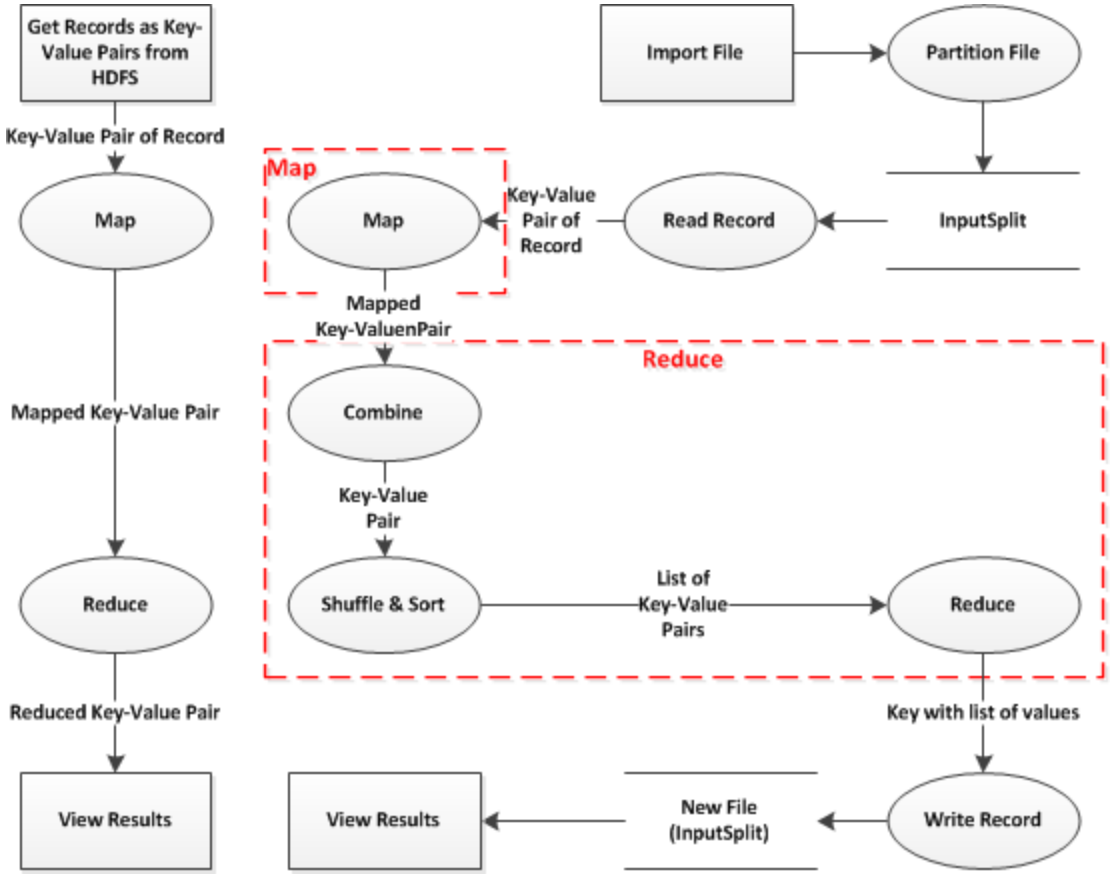
MapReduce (15)



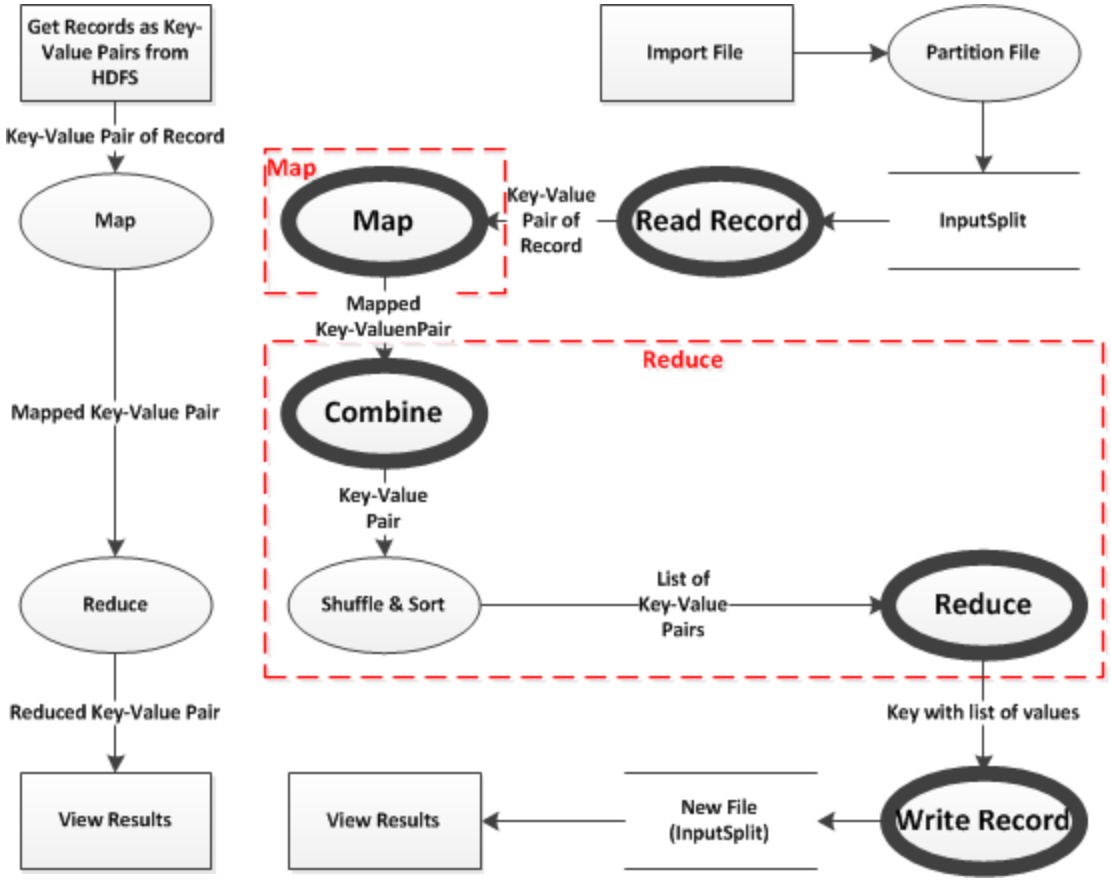
MapReduce (16)



MapReduce (17)



MapReduce (18)



MapReduce (19)

- Record Read
  - Record → (K1,V1)
  - to be or not to be → (0,“to be or not to be”)
- Map
  - (K1,V1) → list(K2,V2)
  - (0,“to be or not to be”) → [(to,1),(be,1),(or,1),(not,1),(to,1),(be,1)]
- Shuffle and Sort
  - list(K2,V2) → (K2,list(V2))
  - [(to,1),(be,1),(or,1),(not,1),(to,1),(be,1)] → (to,[1,1]),(be,[1,1]),(or,[1]),(not,[1])
- Reduce
  - (K2,list(V2)) → list(K3,V3)
  - (to,[1,1]),(be,[1,1]),(or,[1]),(not,[1]) → [(to,2),(be,2),(or,1),(not,1)]
- Record Write
  - (K3,V3) → Records
  - [(to,2),(be,2),(or,1),(not,1)] →

to,2
be,2
or,1
not,1

Break

MapReduce Lab (1)

- Open Eclipse by clicking on the and navigate to WordCount to see the java files that are part of :
  - wordcount\src\solution\
    - WordCount.java
    - WordMapper.java
    - SumReducer.java

- Study these files

- In the terminal:
  - `$ cd ~/workspace/wordcount/src`
  - `$ ls`
  - `$ ls solution`
  - `$ hadoop classpath`
  - `$ javac -classpath `hadoop classpath` solution/*.java`

Note that the apostrophe in this statement is the not the straight or slanted quote (' or ‘) but rather this character (`):  javac -classpath `hadoop classpath` solution/*.java

MapReduce Lab (2)

- In the terminal:
  - `$ jar cvf wc.jar solution/*.class`
  - `$ hadoop jar wc.jar solution.WordCount shakespeare wordcounts`
  - `$ hadoop fs -ls wordcounts`
  - `$ hadoop fs -cat wordcounts/part-r-00000 | less`
  - `$ hadoop jar wc.jar solution.WordCount shakespeare/poems pwords`
  - `$ hadoop fs -rm -r wordcounts pwords`

MapReduce Lab (3)

- Open Firefox
- Start Hue
- Start File Browser in Hue
  - Find wordcounts/part-r-00000
  - View Contents of part-r-00000

C'mon, Hadoop — it's bedtime!

Hadoop

Copyright 2015 by Ernst Henle