CSC 555 and DSC 333
Mining Big Data
Lecture 4

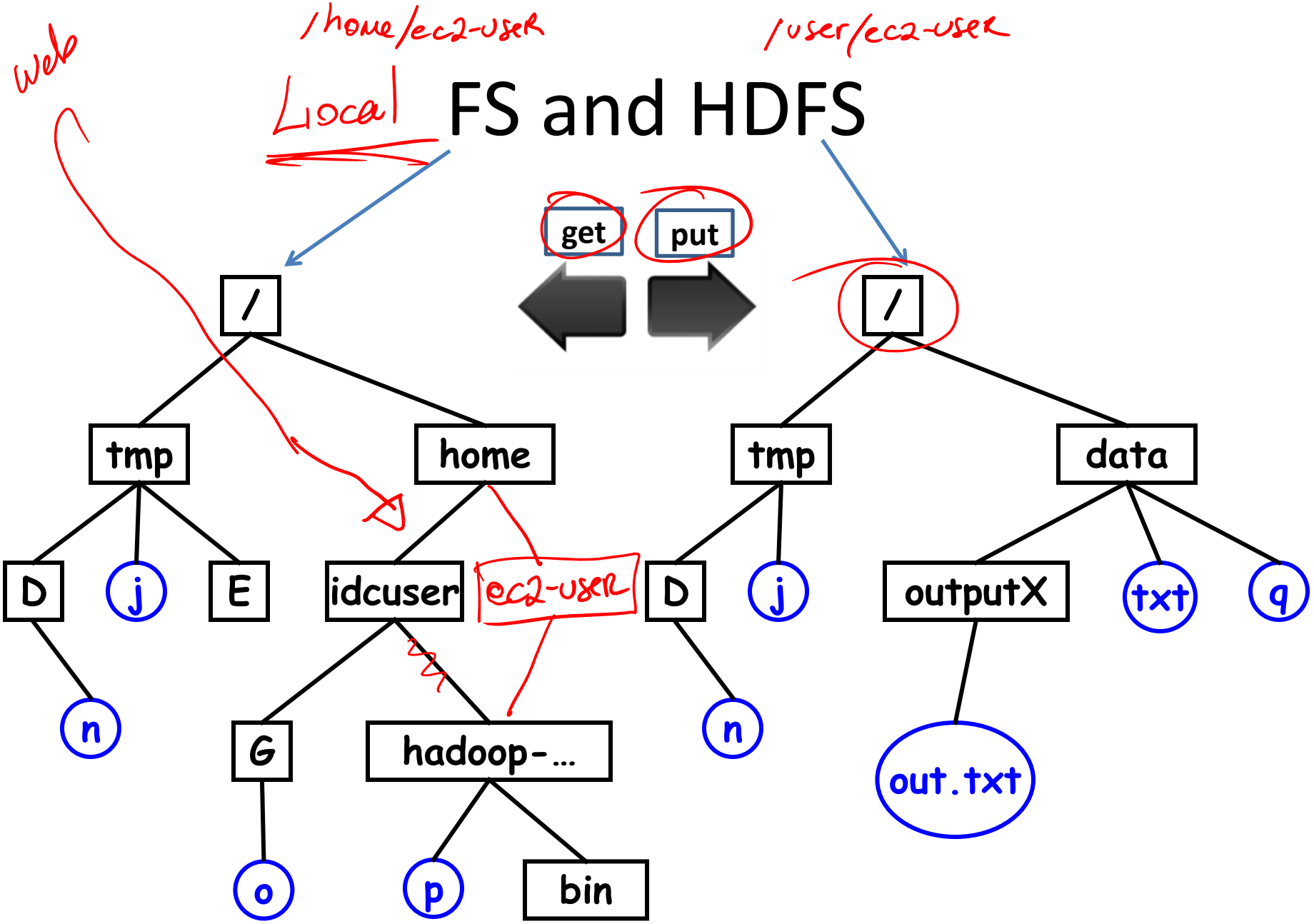
Alexander Rasin
College of CDM, DePaul University
October 5th, 2021

Tonight

- Hadoop/File systems processing
- Hive
  - SELECT TRANSFORM
- Pig
- Hadoop Streaming

Mapper/Reducer in python

FS and HDFS


web
/home/ec2-user
/user/ec2-user
Local
get
put
/
/
tmp
home
tmp
data
D
j
E
idcuser
ec2-user
D
j
outputX
txt
q
n
G
hadoop-...
n
out.txt
o
p
bin

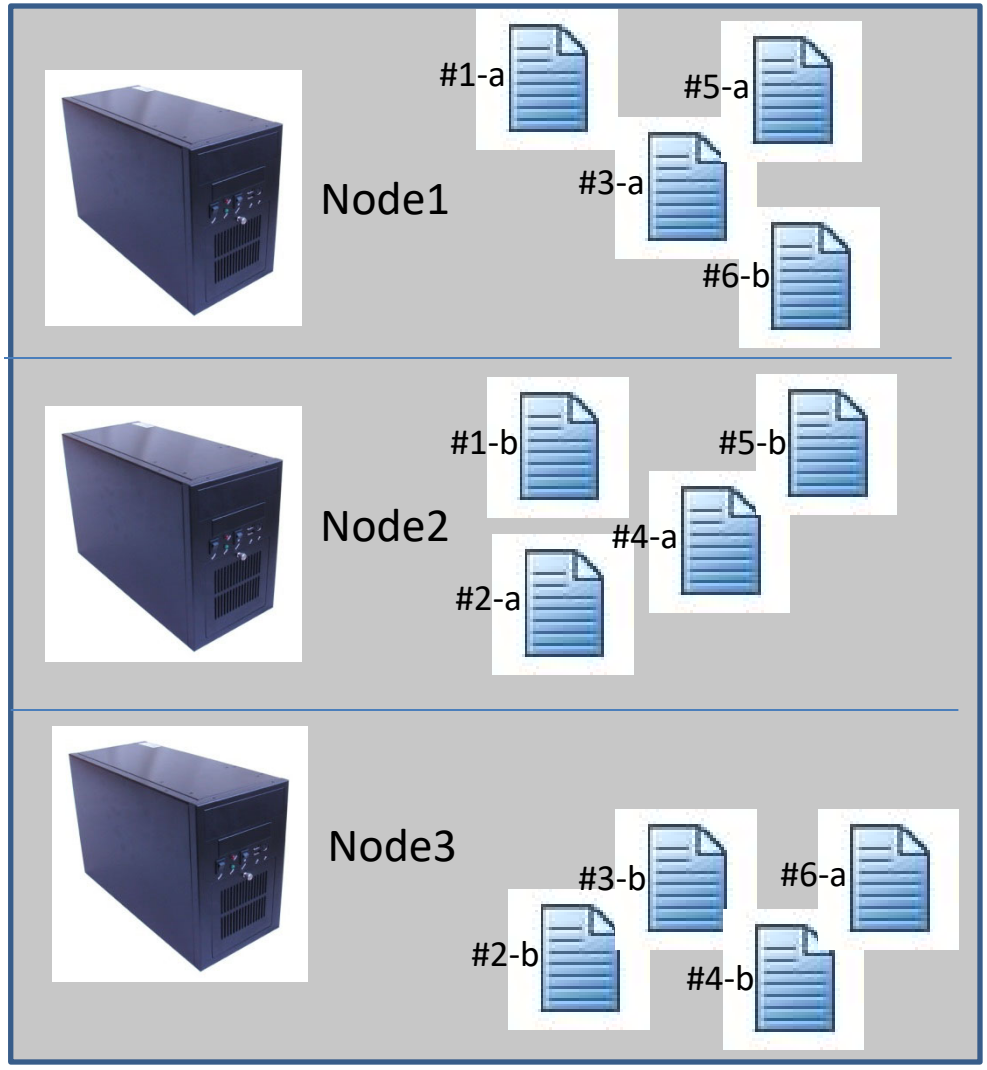
Linux (vs Windows)

- Root directory
  - / (think C:\ on Windows)
- Relative and absolute path
  - /bin vs bin
- Shortcuts
  - . means current directory
  - .. means directory one level up

Linux / Shell setup

- Variables (e.g., HADOOP_HOME)
  - Permanently set in ~/.bashrc (or ~/.bash_profile)
  - Check for settings using "env"

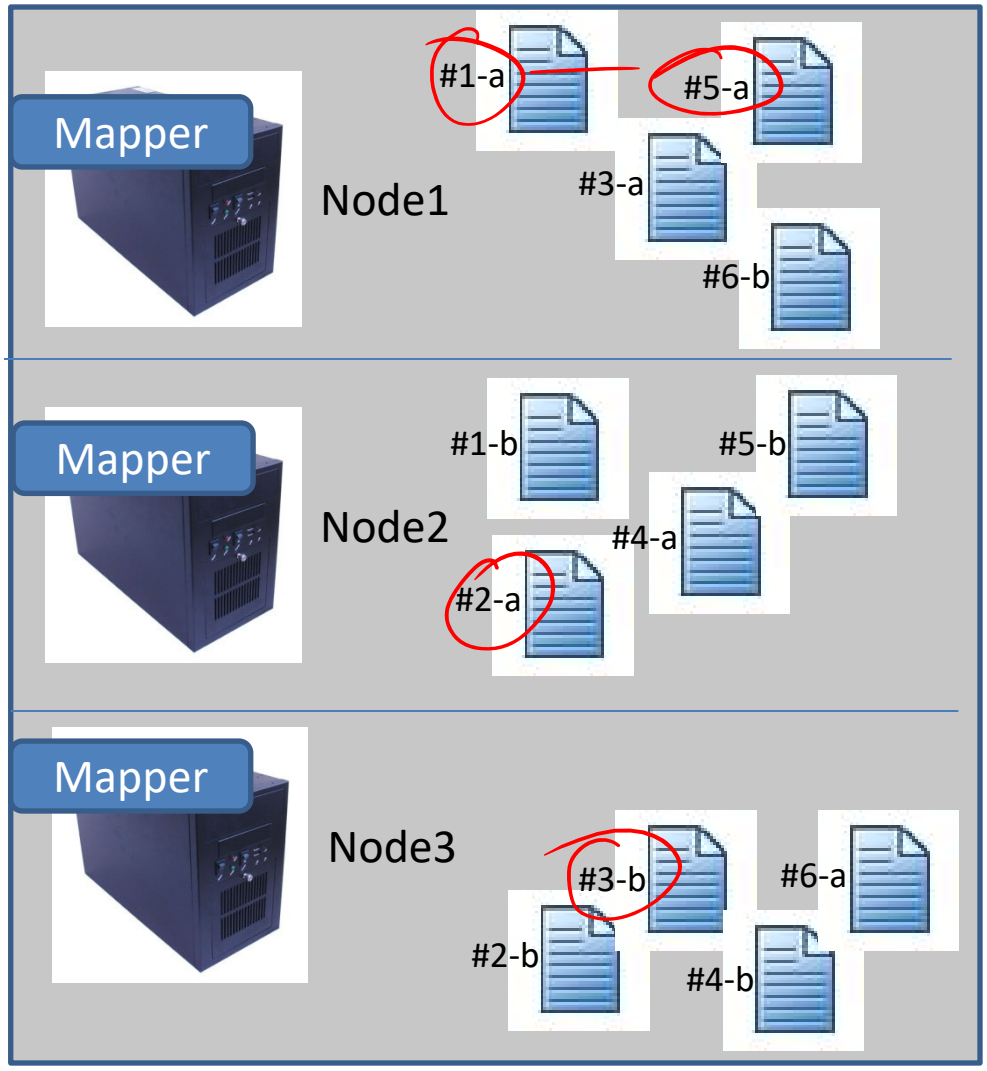
/data/text.txt = 6 X  blocks



NameNode: /data/text.txt, 6 blocks
Block #1 – { Node1, Node2 }
Block #2 – { Node2, Node3 }
…
Block #6 – { Node3, Node1 }

Job: Perform wordcount
(a.k.a. word distribution)

/data/text.txt = 6 X  blocks



NameNode: /data/text.txt, 6 blocks
Block #1 – { Node1, Node2 }
Block #2 – { Node2, Node3 }
…
Block #6 – { Node3, Node1 }

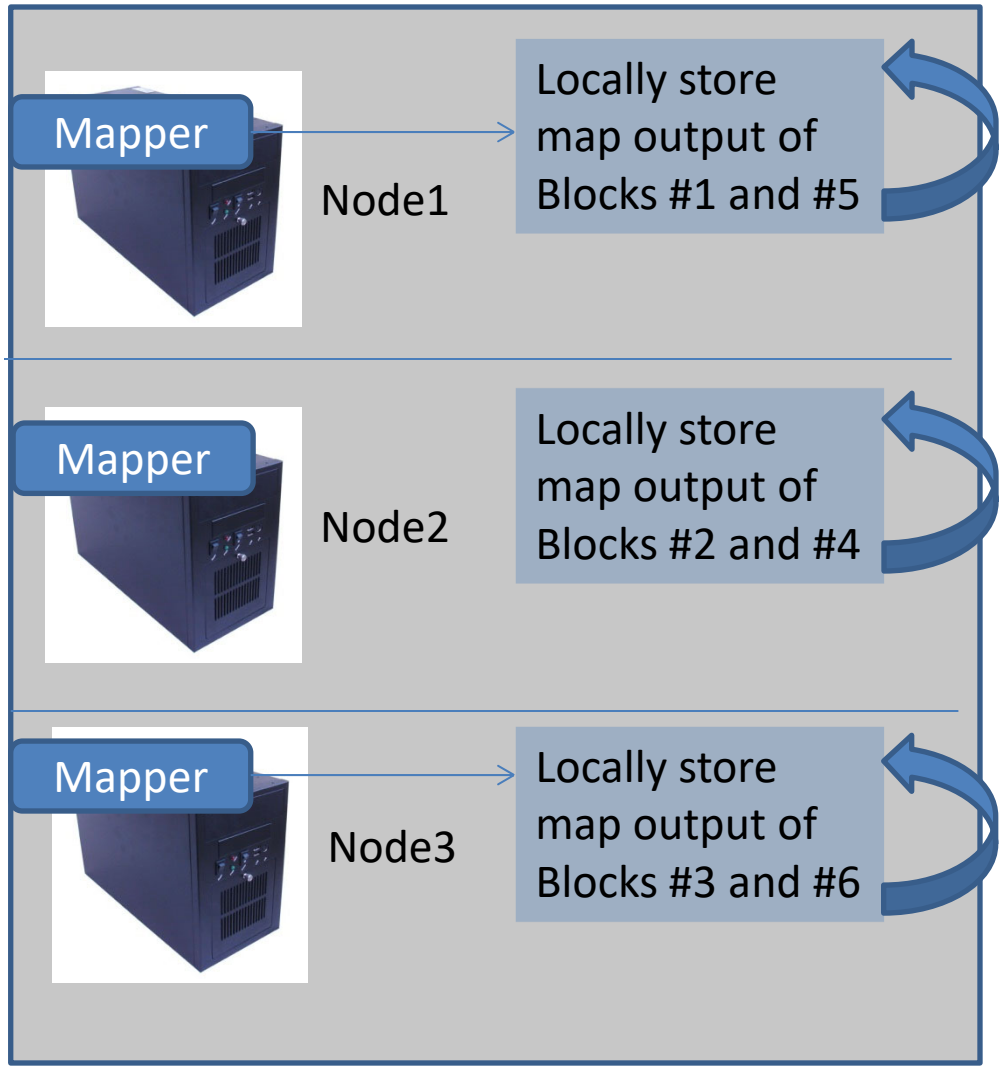
JobTracker:

Step 1: Talk to the NameNode

Step 2: Identify TaskTrackers with available slots (near data)

Wait/monitor TaskTrackers

/data/text.txt = 6 X  blocks

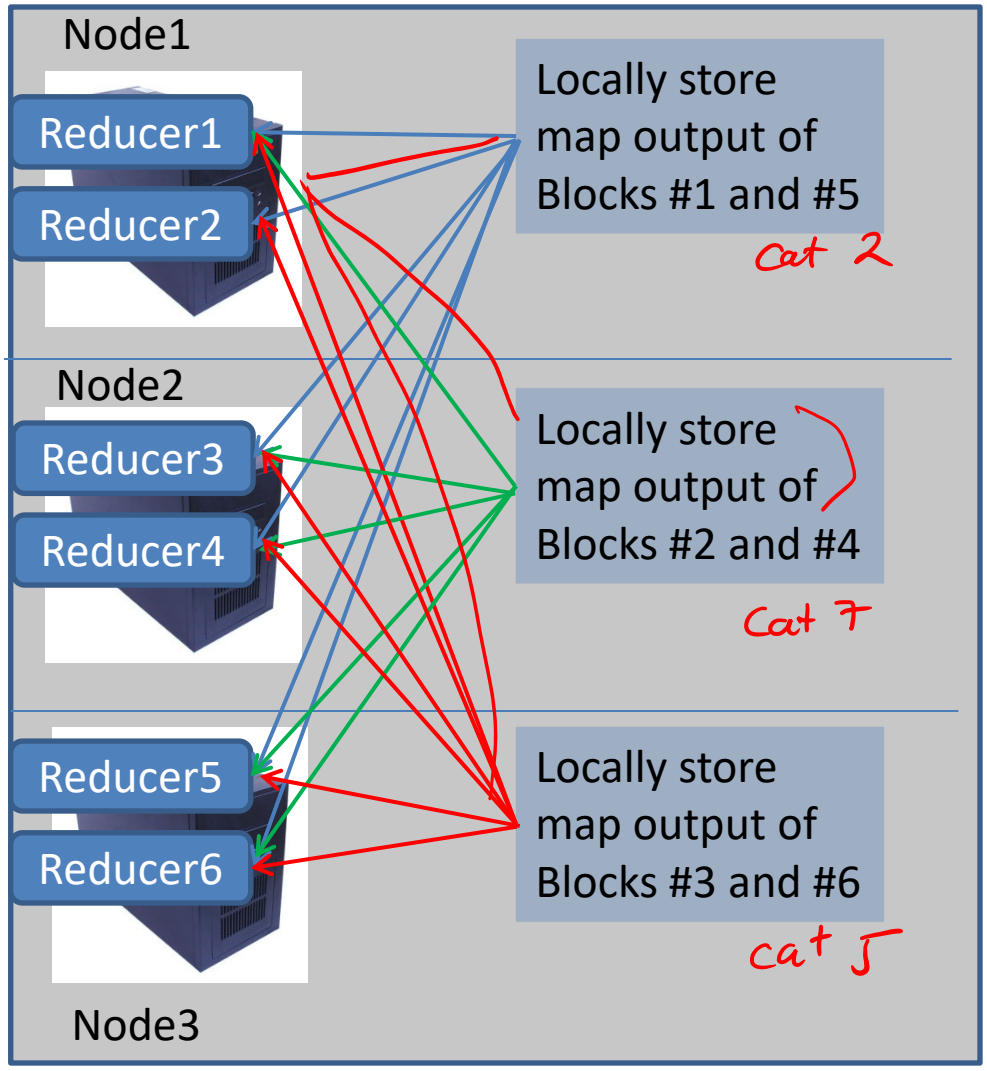


Cost: Read and process 2 blocks on each node (plus write a local copy)

NOTE: Local map output could be larger (e.g., 3 blocks)

Optional: Apply a Combiner to the locally stored output

/data/text.txt = 6 X  blocks



Cost: Send each key to the respective reducer

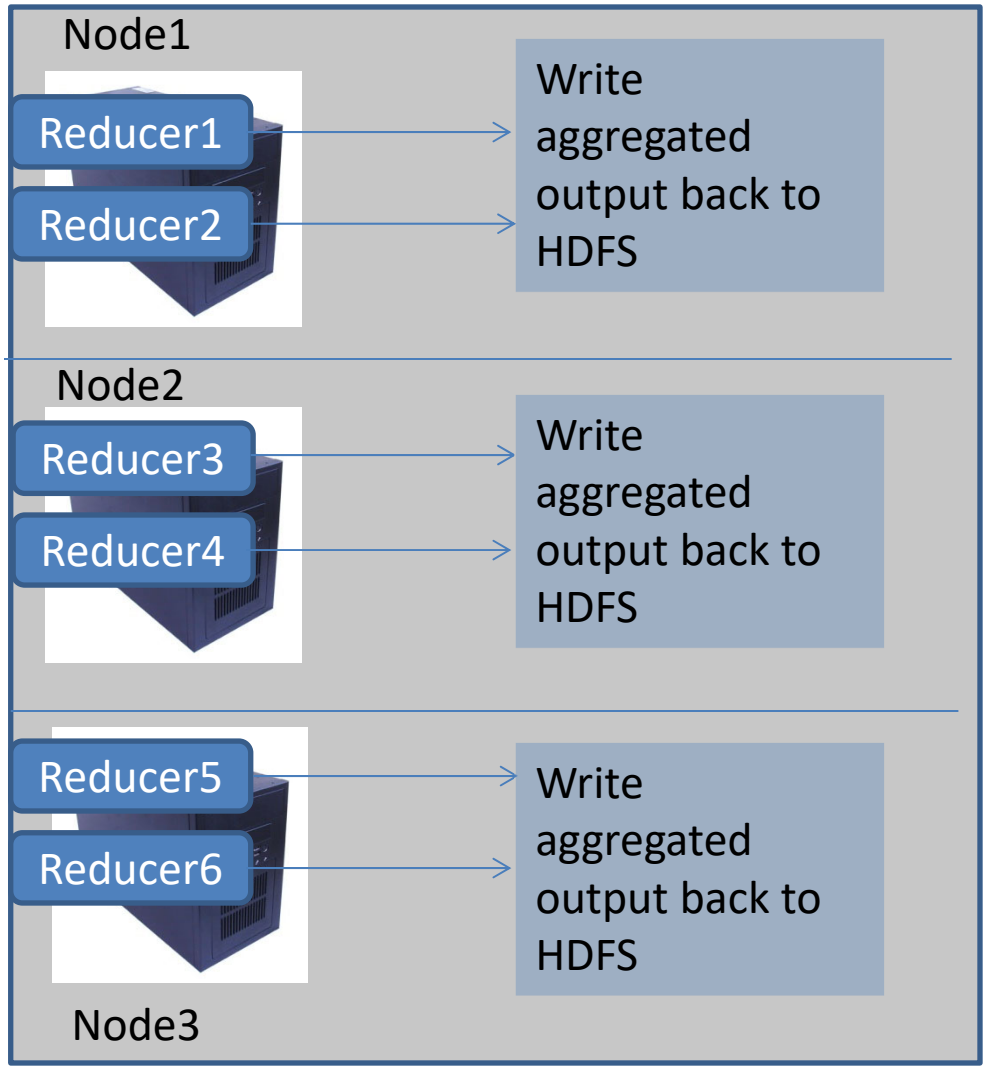
NOTE: Over the network transfer (slow).

Keys arrive sorted:

Reducer1: (a {1, 4, 5})
(cat {2, 5, 7})
(zing {1, 2, 1})

Reducer5: (bat {2, 1})
(pie {1, 1}}
(young {1, 8})

/data/text.txt = 6 X  blocks



Cost: Process the aggregation and write to HDFS (2X replication)

One file per reducer (outputs.per.reducer can be set to generate several files per reducer)

Hive Setup

CREATE TABLE u_data (userid INT,

movieid INT,

rating INT,

unixtime STRING)

ROW FORMAT DELIMITED FIELDS

TERMINATED BY '\t' STORED AS TEXTFILE; (not compressed)

- show tables; describe u_data;
- wget http://www.grouplens.org/system/files/ml-100k.zip

LOAD DATA **LOCAL** INPATH 'ml-100k/u.data'

OVERWRITE INTO TABLE u_data;


u_data
userid
movieid
rating
timestamp
u_data_new
userid
movieid
rating
dayoftheweek

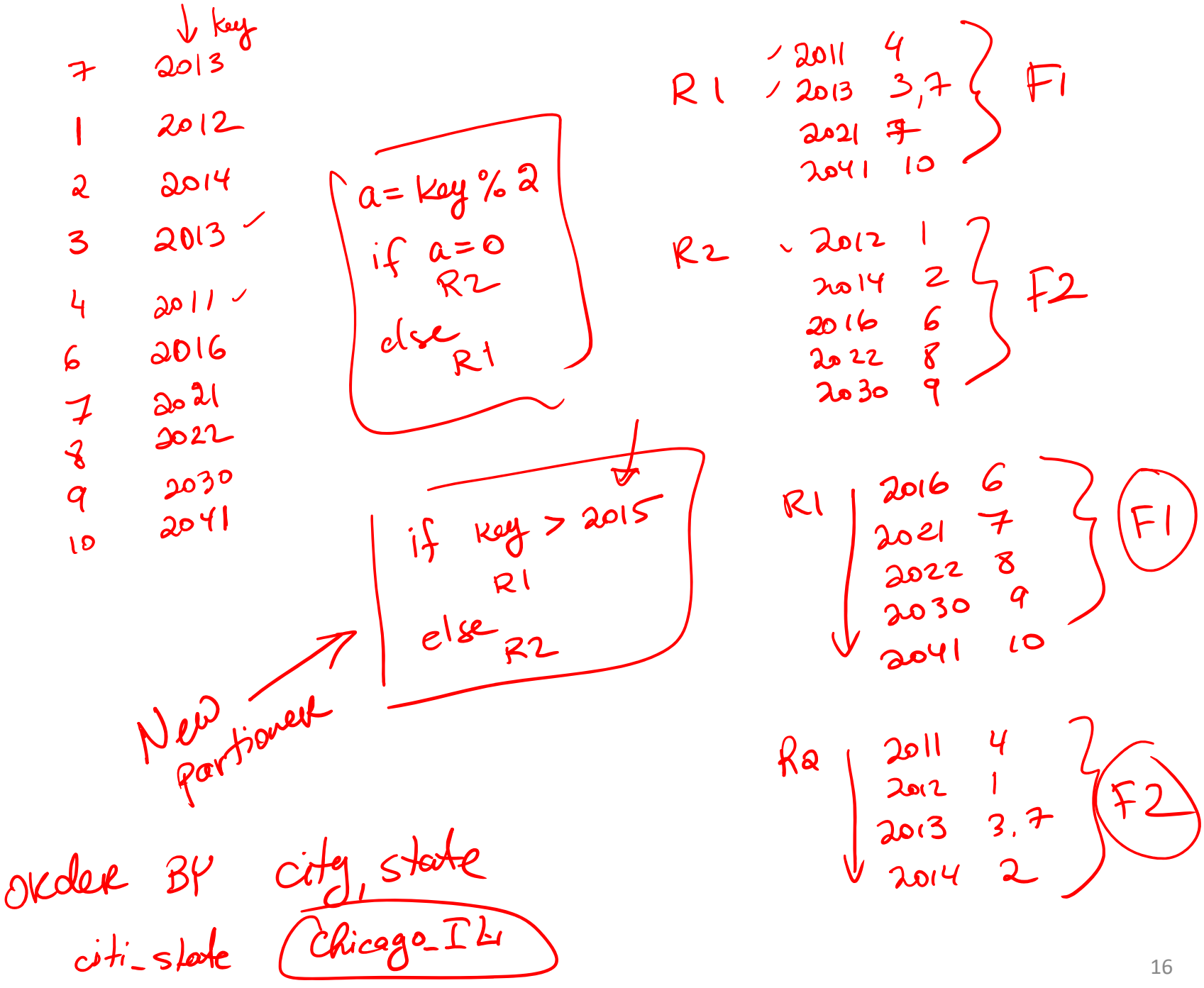
Custom Hive Transformation

```
CREATE TABLE u_data_new ( userid INT,
movieid INT, rating INT, weekday String)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t';
add FILE weekday_mapper.py;
INSERT OVERWRITE TABLE u_data_new
SELECT TRANSFORM (userid, movieid, rating, unixtime)
USING 'python weekday_mapper.py'
AS (userid, movieid, rating, weekday) FROM u_data;
```

A Break 7:35

Using Hive

```
SELECT weekday, COUNT(*)
FROM u_data_new GROUP BY weekday;

SELECT weekday, COUNT(*)  as Total
FROM u_data_new GROUP BY weekday
ORDER BY Total;
```


key
7 2013
1 2012
2 2014
3 2013
4 2011
6 2016
7 2021
8 2022
9 2030
10 2041
a = key % 2
if a = 0
R2
else
R1
R1
2011 4
2013 3,7
2021 7
2041 10
F1
R2
2012 1
2014 2
2016 6
2022 8
2030 9
F2
if key > 2015
R1
else
R2
New partioner
R1
2016 6
2021 7
2022 8
2030 9
2041 10
F1
R2
2011 4
2012 1
2013 3,7
2014 2
F2
Order BY city, state
citi_state
Chicago_IL

MapReduce Sort

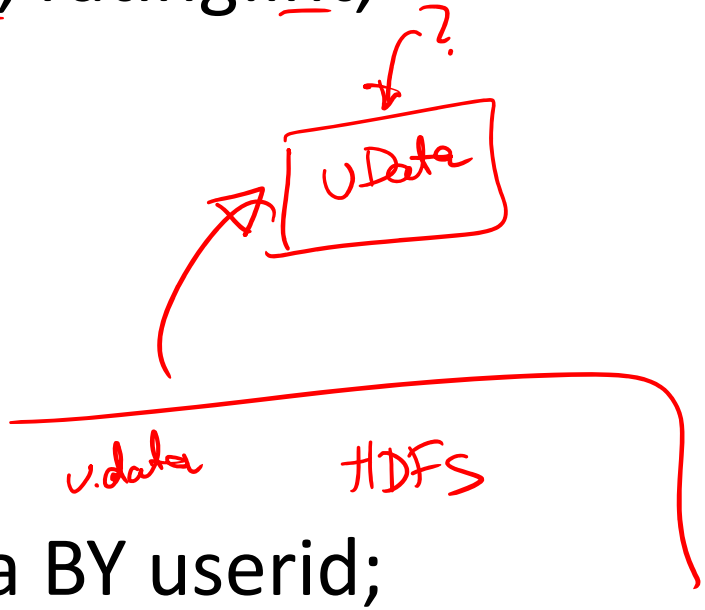
- Distributed sort operation
- Caveats
  - Can only sort by the key
  - Keys sorted within reducer
  - Special partitioner

Hive Partitioning

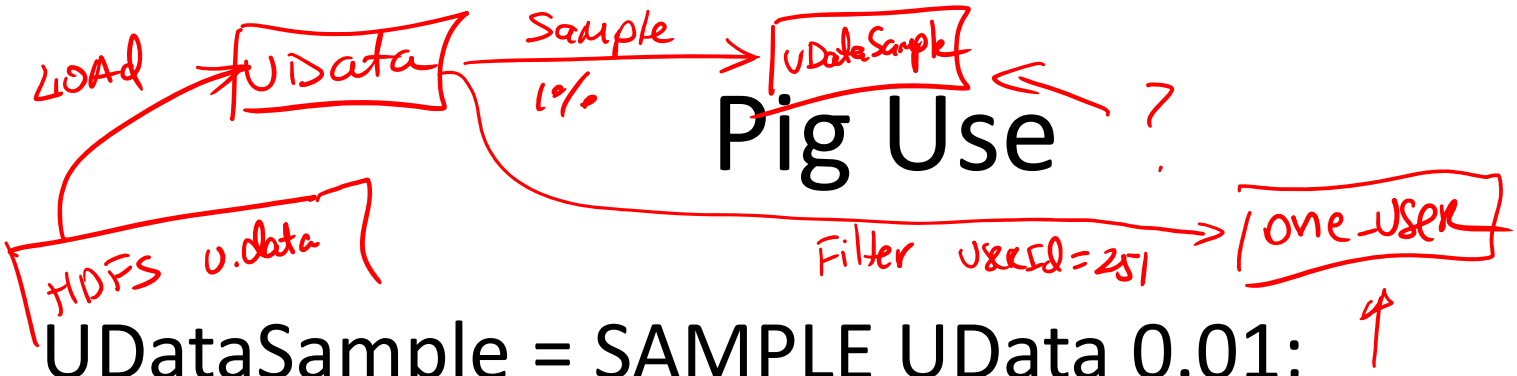
```
CREATE TABLE u_data ( userid INT,
movieid INT, rating INT, unixtime STRING)
PARTITIONED BY(country STRING)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH 'ml-100k/u.data'
OVERWRITE INTO TABLE u_data
PARTITION(country='France');
show partitions u_data;
```

Pig Setup

```
UData = LOAD 'u.data' USING PigStorage('\t') AS
(userid:int, movieid:int, rating:int,
unixtime:chararray);

DESCRIBE UData;

DUMP UData;

UDataS = ORDER UData BY userid;

DUMP UDataS;
```

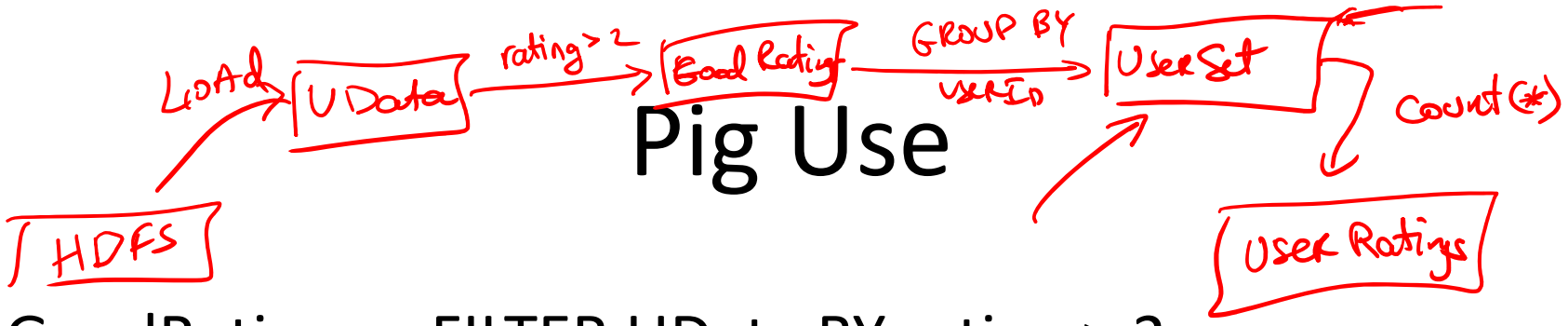


Pig Use




```
UDataSample = SAMPLE UData 0.01;
DUMP UDataSample;
STORE UDataSample INTO 'UDataSample'
USING PigStorage ('|');

ONE_USER = FILTER UData BY userid == 251;
DUMP ONE_USER;
```

Pig Use




```
GoodRatings = FILTER UData BY rating > 2;
UserSet = GROUP GoodRatings BY userid;
DUMP UserSet;
UserRatings = FOREACH UserSet GENERATE
COUNT(GoodRatings);
DUMP UserRatings;
ILLUSTRATE UserRatings;
```

```
Select Count(*)
FROM UData
Where rating > 2
GROUP BY userid
```

Pig Use

```
UserSet2 = GROUP UData BY userid;
UserRatings2 = FOREACH UserSet2 GENERATE
UData.userid, AVG(UData.rating);
DUMP UserRatings2;
```

Pig Use

```
UDataNew = LOAD 'u.data.new' USING PigStorage('\t') AS
(userid2:int, movieid2:int, rating2:int, weekday2:chararray);

DESCRIBE UDataNew; DUMP UDataNew;

DataJoin=JOIN UData BY (userid, movieid), UDataNew BY
(userid2, movieid2);

FilterDataJoin = FILTER DataJoin BY rating != 2;

FDL_LG = GROUP FilterDataJoin BY (UDataNew::userid2,
UDataNew::rating2);

Result = FOREACH FDL_LG GENERATE group,
COUNT(FilterDataJoin);
```

FROM UData, UDataNew
where userid = userid2
AND movieid = movieid2

Pig Use

UDataNew = LOAD 'u.data.new' USING PigStorage('\t') AS (userid2:int, movieid2:int, rating2:int, weekday2:chararray);

DESCRIBE UDataNew; DUMP UDataNew;

DataJoin= JOIN UData BY ($0,$1),UDataNew BY ($0,$1);

STORE DataJoin INTO 'DataJoin' using PigStorage(',');

DataJoinL = LOAD 'DataJoin' USING PigStorage(',') AS (uid1:int, mid1:int, r1:int, ut1:chararray, uid2:int, mid2:int, r2:int, wd2:chararray);

DataJoinLG = GROUP DataJoinL BY wd2;

DataJoinG = ORDER (FOREACH DataJoinLG GENERATE group, AVG(DataJoinL.r1)) BY group;

Pig Use

- cd hdfs:///
- ls
- mkdir testpig
- copyFromLocal /etc/passwd passwd
- passwd = LOAD 'hdfs:///testpig/passwd' USING PigStorage(':') AS (user:chararray, passwd:chararray, uid:int, gid:int, userinfo:chararray, home:chararray, shell:chararray);

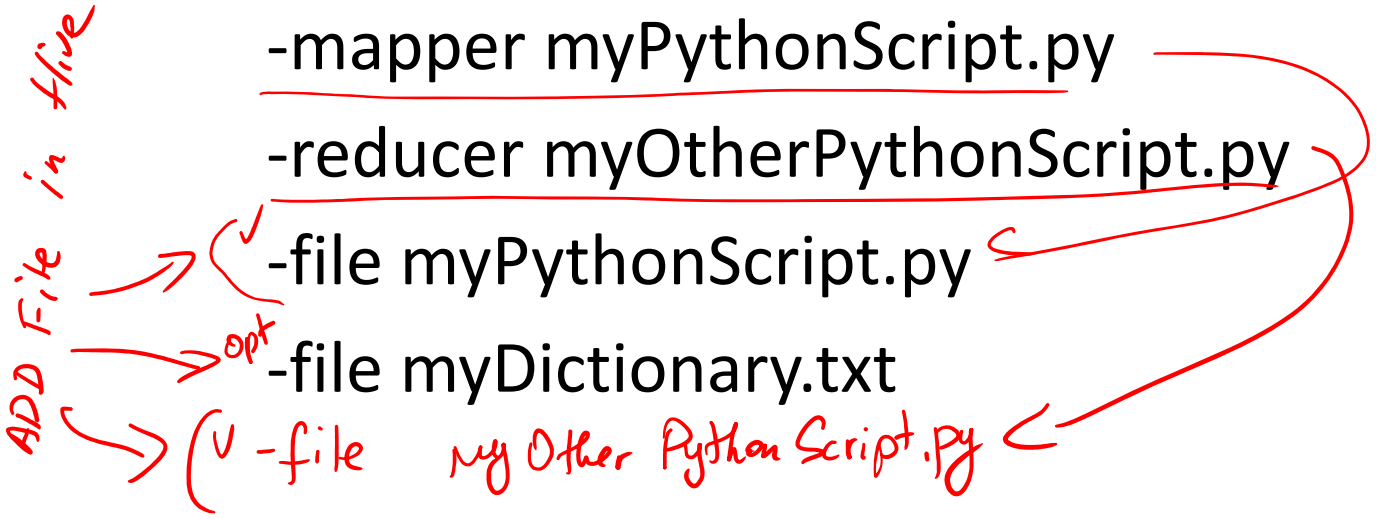
Pig Use

- DUMP passwd;
- groupShell = GROUP passwd BY shell;
- DUMP groupShell;

- groupCount = FOREACH groupShell GENERATE group, COUNT(passwd);

Hadoop Streaming: The Idea

```
hadoop jar  hadoop-streaming.jar
    -input myInputDirs
    -output myOutputDir
    -mapper myPythonScript.py
    -reducer myOtherPythonScript.py
    -file myPythonScript.py
    -file myDictionary.txt
```



(WE STOPPED HERE)
Hadoop Streaming

- Map: cat binary
- Reduce: value count in python
- hadoop jar hadoop-streaming-2.6.4.jar -input /home/ec2-user/mlens -output /data/output1 -mapper /bin/cat -reducer myReducer.py -file myReducer.py

Hadoop Streaming

- Map: Timestamp=>Weekday in python
- Reduce: value count in python
- hadoop jar hadoop-streaming-2.6.4.jar -input /home/ec2-user/mlens -output /data/output4 -mapper myMapper.py -reducer myReducer.py -file myReducer.py -file myMapper.py

Hadoop Streaming Options

- -D stream.map.output.field.separator=|
- -mapper org.apache.hadoop.mapred.lib.IdentityMapper
- -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner
- -D mapred.reduce.tasks=2
- -D mapred.text.key.comparator.options=-nr

Hadoop Streaming

- Map: cat binary
- Reduce: value count in python
- hadoop jar hadoop-streaming-2.6.4.jar -D mapred.reduce.tasks=3 -D mapred.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedComparator -D mapred.text.key.comparator.options=-nr -input /home/ec2-user/mlens -output /data/output3 -mapper /bin/cat -reducer myReducer.py -file myReducer.py

MapReduce Debuging

- !#/usr/bin/python
- Map code:
  - cat test.txt | python myMapper.py
- Shuffle code:
  - cat test.txt | python myMapper.py | sort
- Reduce code:
  - cat test.txt | python myMapper.py | sort | python myReducer.py

Hadoop DistributedCache

- Lookup tables
- Data transferred before JVM starts
  - -file myMapper.py
- Hive
  - add FILE weekday_mapper.py;

Next Time:

- Hadoop Streaming
- Building a distributed cluster
- Compression
- NoSQL