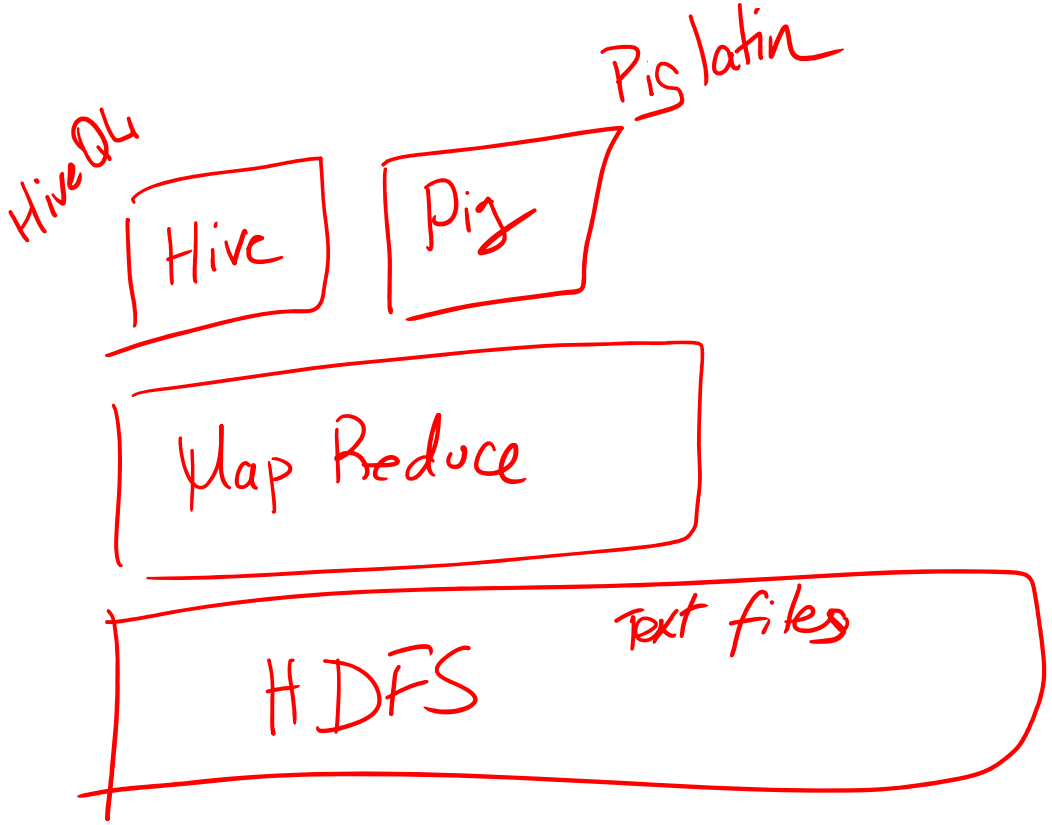
CSC 555 and DSC 333
Mining Big Data
Lecture 5

Alexander Rasin
College of CDM, DePaul University
October 12th, 2021

Tonight

- Star schema / SSBM
- Hive and Pig
- Hadoop Streaming
- HBase


HiveQL
Hive
Pig
Pig latin
Map Reduce
HDFS
Text files

Star-Schema

“Fact” table

Transactions

tid	cid	pid	date	amount
1	3	4	01-01-10	200.00
2	1	10	01-01-10	45.95
3	3	9	01-02-10	110.00

“Dimension” table

Customers

cid	name	address
1	…	
2	…	
3	Alex	

“Dimension” table

Products

pid	name	desc
1		
2		
3		
4	LCD Monitor	…

“Dimension” table

Date

date	day	holiday

SSBM Schema

- 4 Dimension tables
  - Part: manufacturer, name, size, ...
  - Supplier: name, address, phone, …
  - Customer: name, address, …
  - Date: month, year, dayofweek, …
- 1 Fact table: lineorder
  - Composite key (orderkey+linenumber)
  - 4 foreign keys (one per dimension)
  - Orderpriority, quantity, discount, tax, …

Join in Hive

- Hive StarSchema optimization

```
SELECT Address, Count(*)
FROM SmallTable st JOIN
    BigTable bt ON (st.id = bt.id)
GROUP BY Address
```

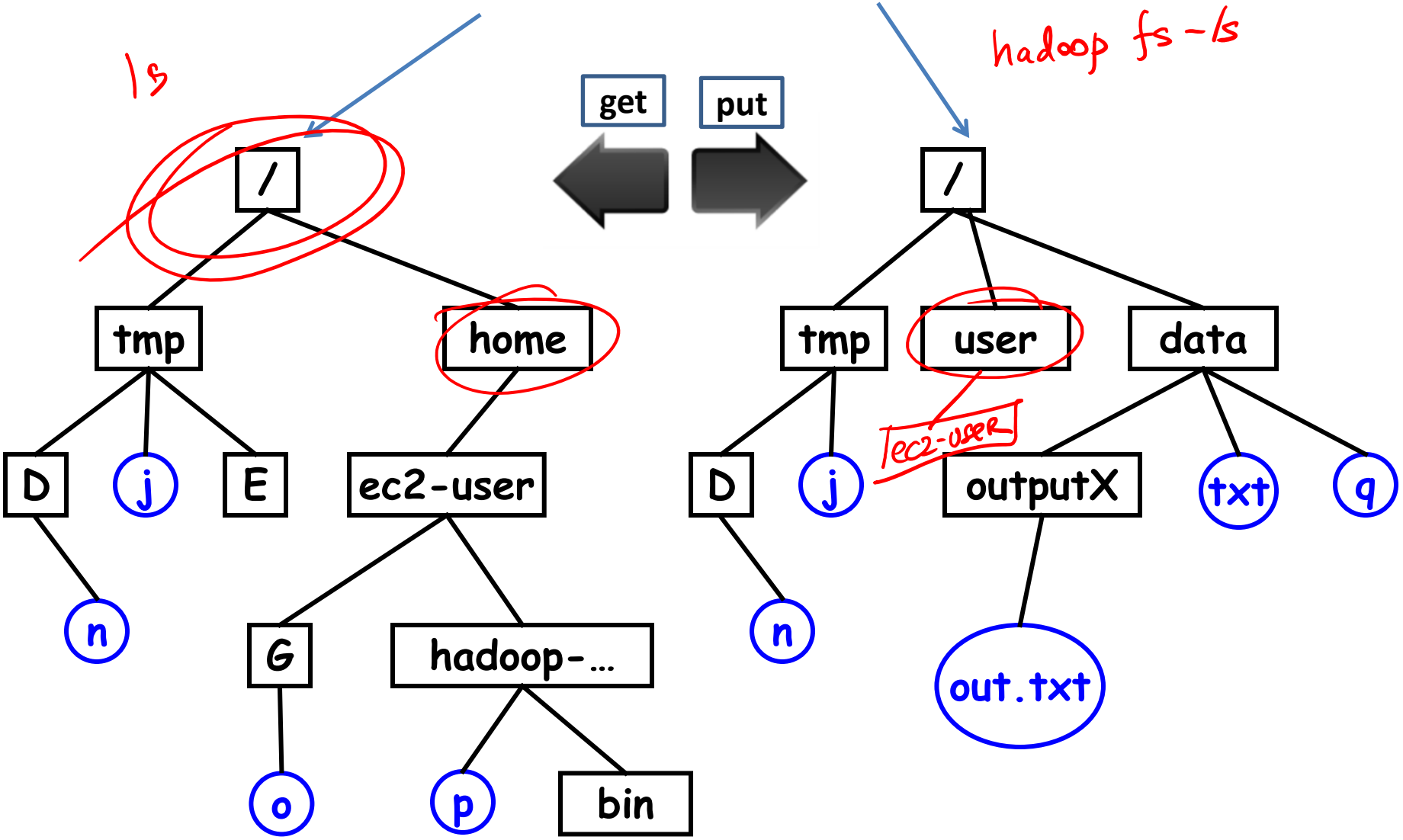
- 2 MapReduce jobs
- Network transfer

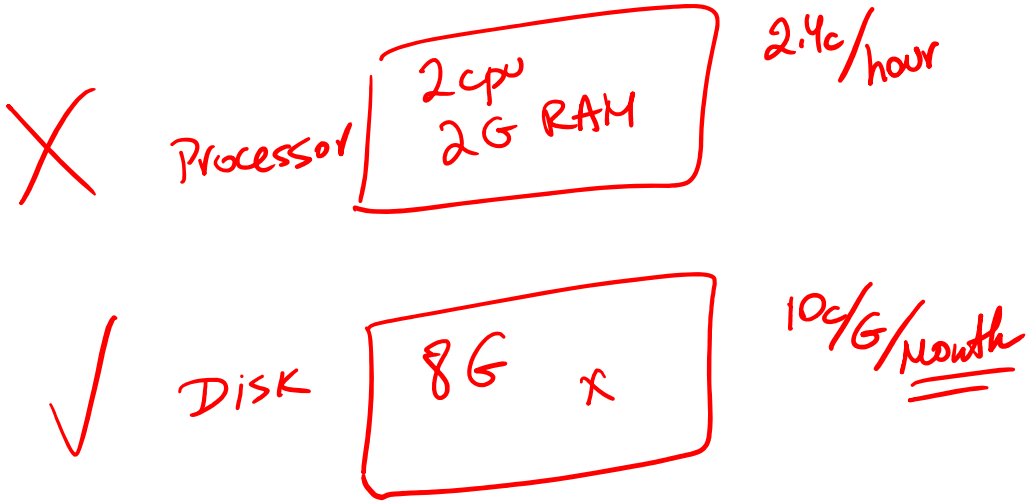
Map-Join

```
SELECT /*+ MAPJOIN(st) */
Address, Count(*)
FROM SmallTable st JOIN
        BigTable bt ON (st.id = bt.id)
GROUP BY Address
```

- Replicate the SmallTable
- set hive.auto.convert.join = true;
  - hive.mapjoin.smalltable.filesize (25MB default)

LocalFS and HDFS




X
Processor
2cpu
2G RAM
2.4c/hour
√
Disk
8G x
10c/G/Month

Pig Example

- 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);

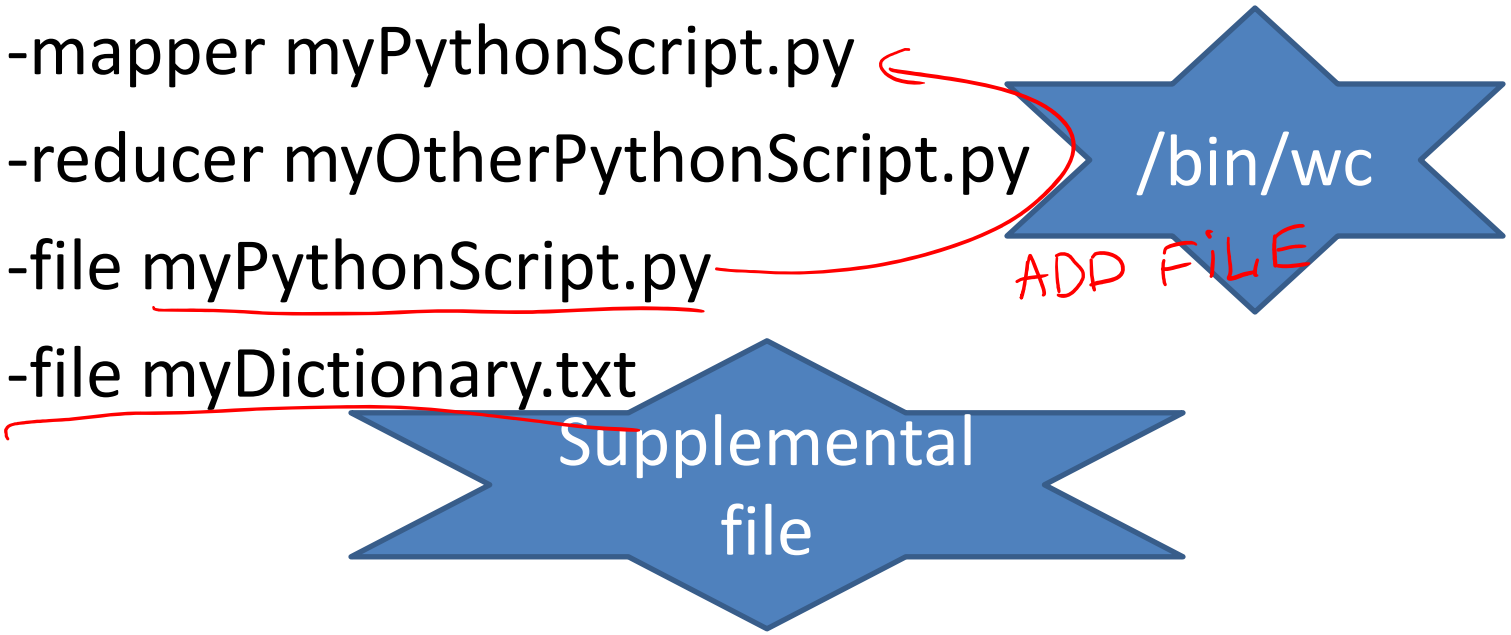
Pig Use

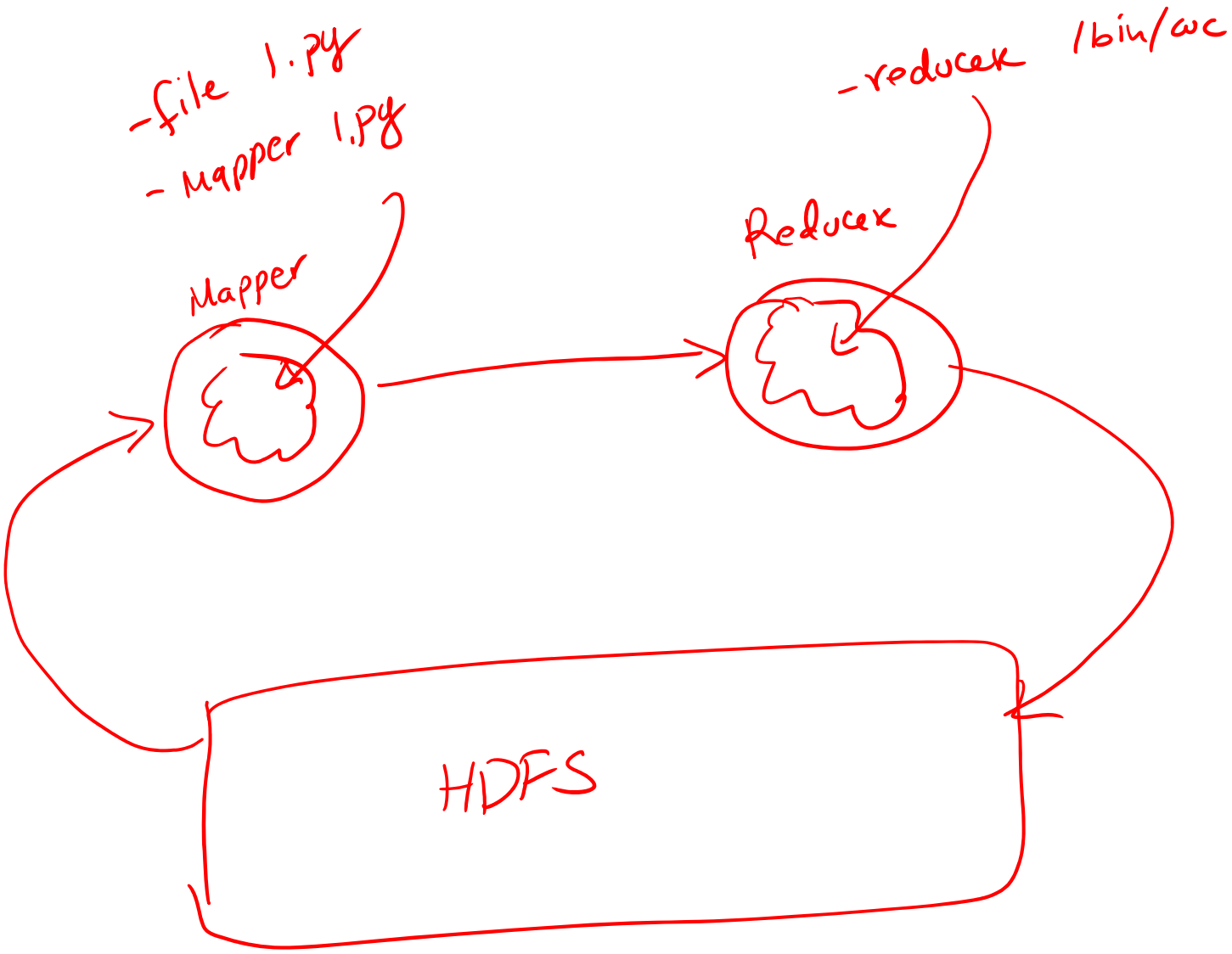
- DUMP passwd;
- groupShell = GROUP passwd BY shell;
- DUMP groupShell;

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

Hadoop Streaming

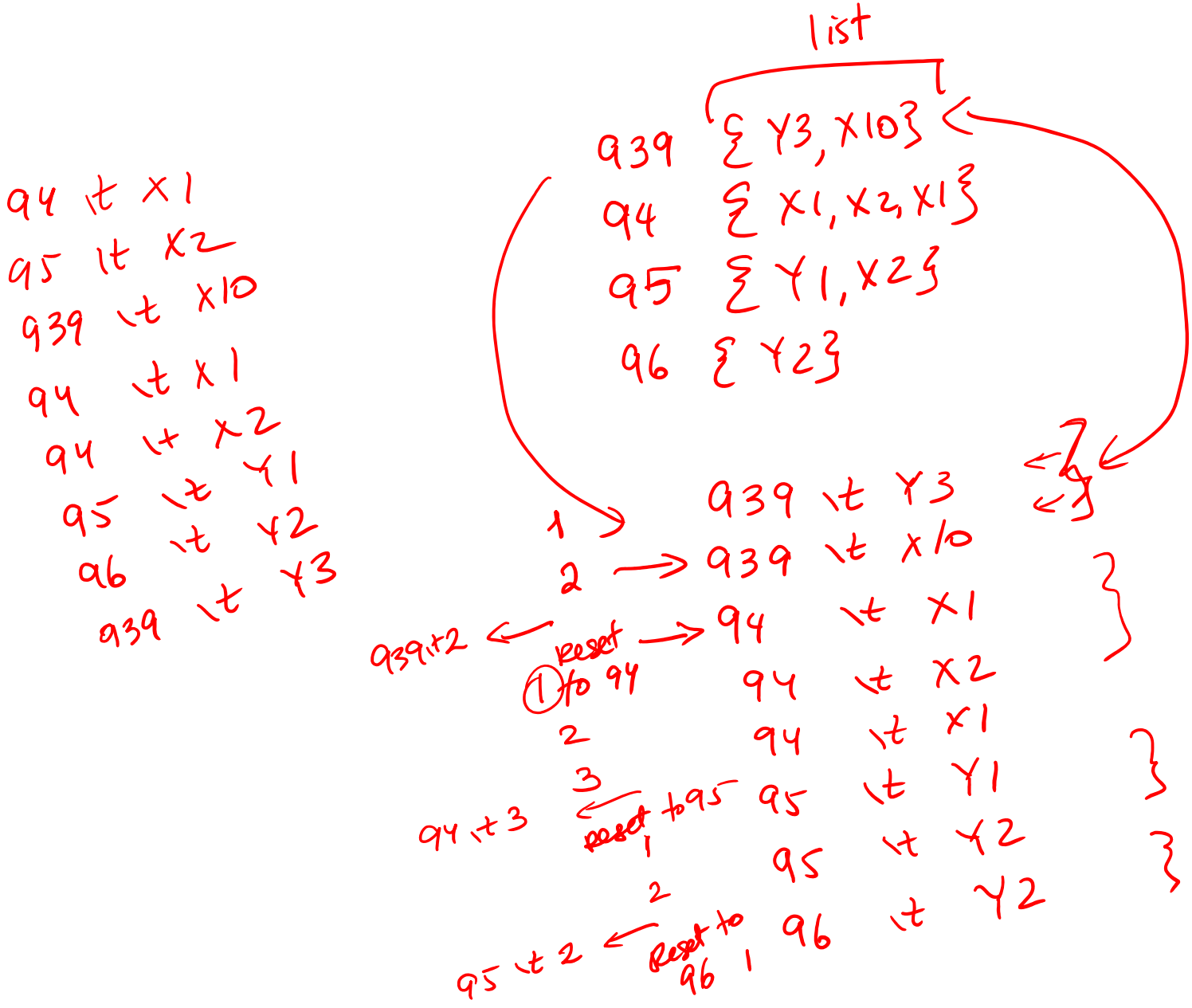
```
hadoop jar  hadoop-streaming.jar
    -input myInputDirs
    -output myOutputDir
    -mapper myPythonScript.py
    -reducer myOtherPythonScript.py
    -file myPythonScript.py
    -file myDictionary.txt
```




-file 1.py
-mapper 1.py
Mapper
-reducer /bin/wc
Reducer
HDFS

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


list
939 { Y3, X10 }
94 { X1, X2, X1 }
95 { Y1, X2 }
96 { Y2 }
94 t X1
95 t X2
939 t X10
94 t X1
94 t X2
95 t Y1
95 t Y2
96 t Y3
939 t Y3
939 t X10
94 t X1
94 t X2
94 t X1
95 t Y1
95 t Y2
96 t Y2
1
2
939 t2
reset to 94
2
3
94 t3
reset to 95
1
2
95 t2
reset to 96
1

Hadoop Streaming

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


7:42

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

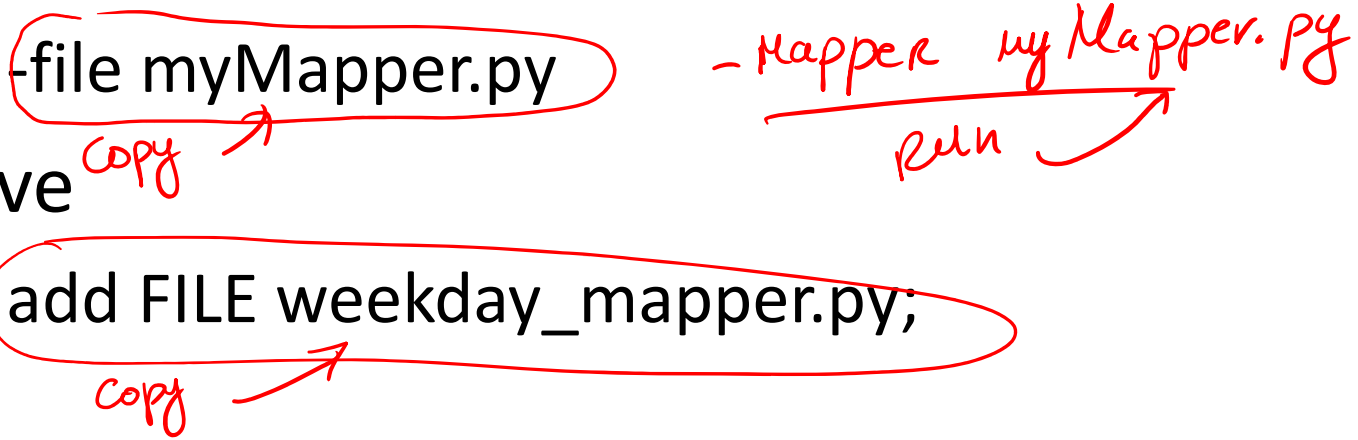
- Map: cat binary
- Reduce: value count in python
- hadoop jar hadoop-streaming-2.6.4.jar -D mapred.reduce.tasks=3 -D mapred.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedComparator -D mapred.text.key.comparator.options=-n -input /user/ec2-user/mlens -output /data/output3 -mapper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

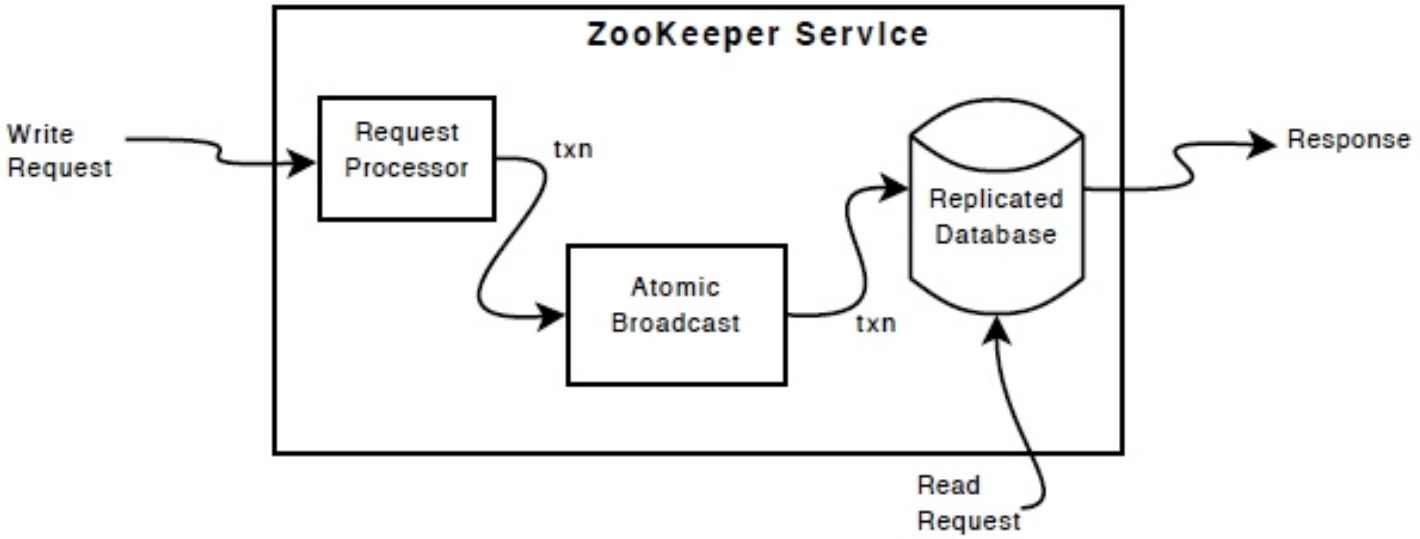
- Lookup tables
- Data transferred before JVM starts
  - -file myMapper.py
- Hive
  - add FILE weekday_mapper.py;

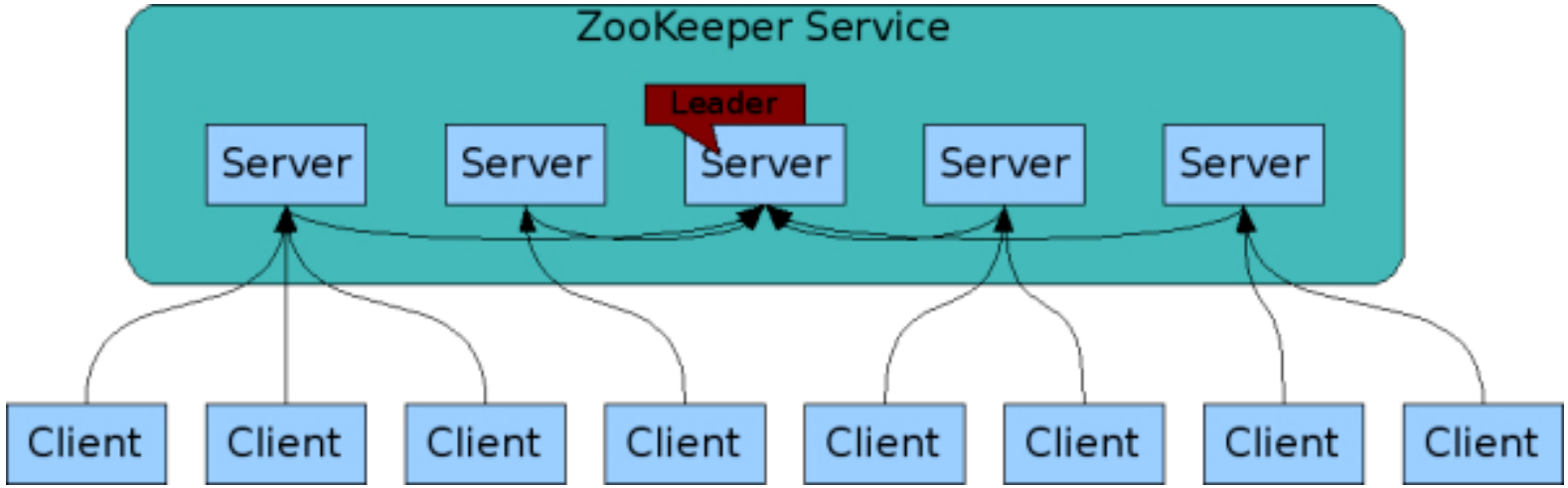


Zookeeper

- “Because coordinating distributed systems is a zoo”
- Coordinate service for distributed applications
  - Naming
  - Configuration
  - Synchronization
  - Group management/leader election

Zookeeper Overview





Zookeeper Uses

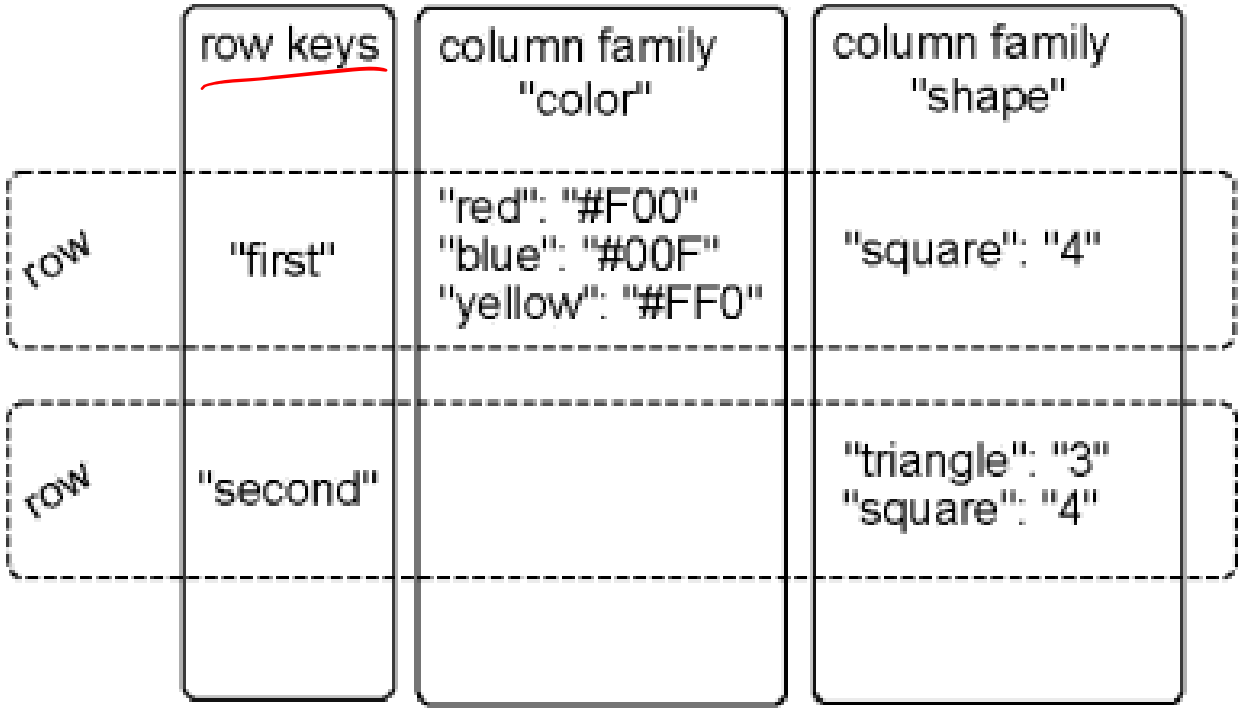
- Barriers
  - Delay processing until a condition is met
- Queues
  - (priority) Distributed queue
- Locks
  - Exclusive access to data
- Leader election
  - Select one “leader” node
  - Find a replacement when needed

Wide-Column Stores

Keys and Values

Optimize column family distribution

Add columns (no penalty)



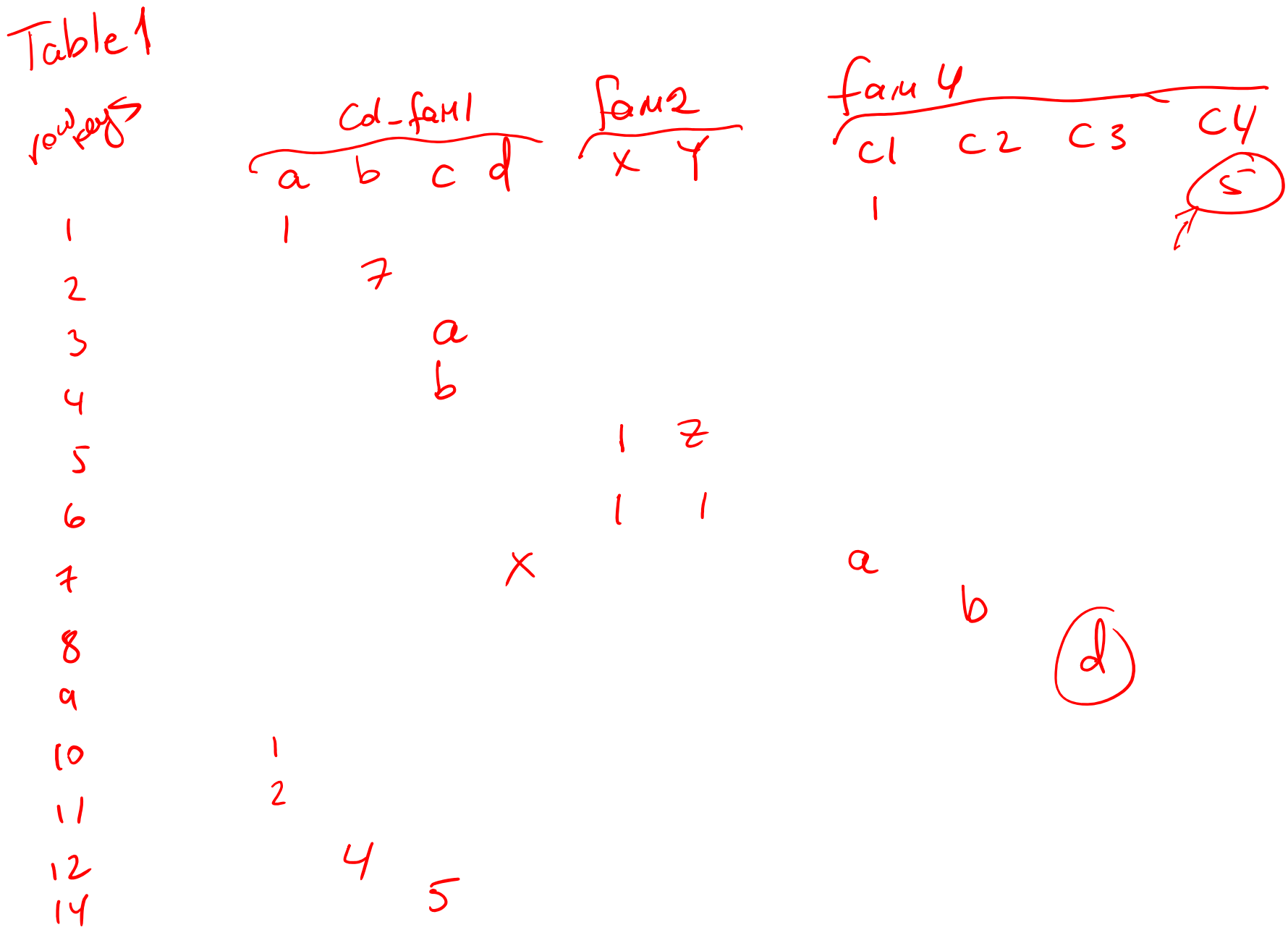
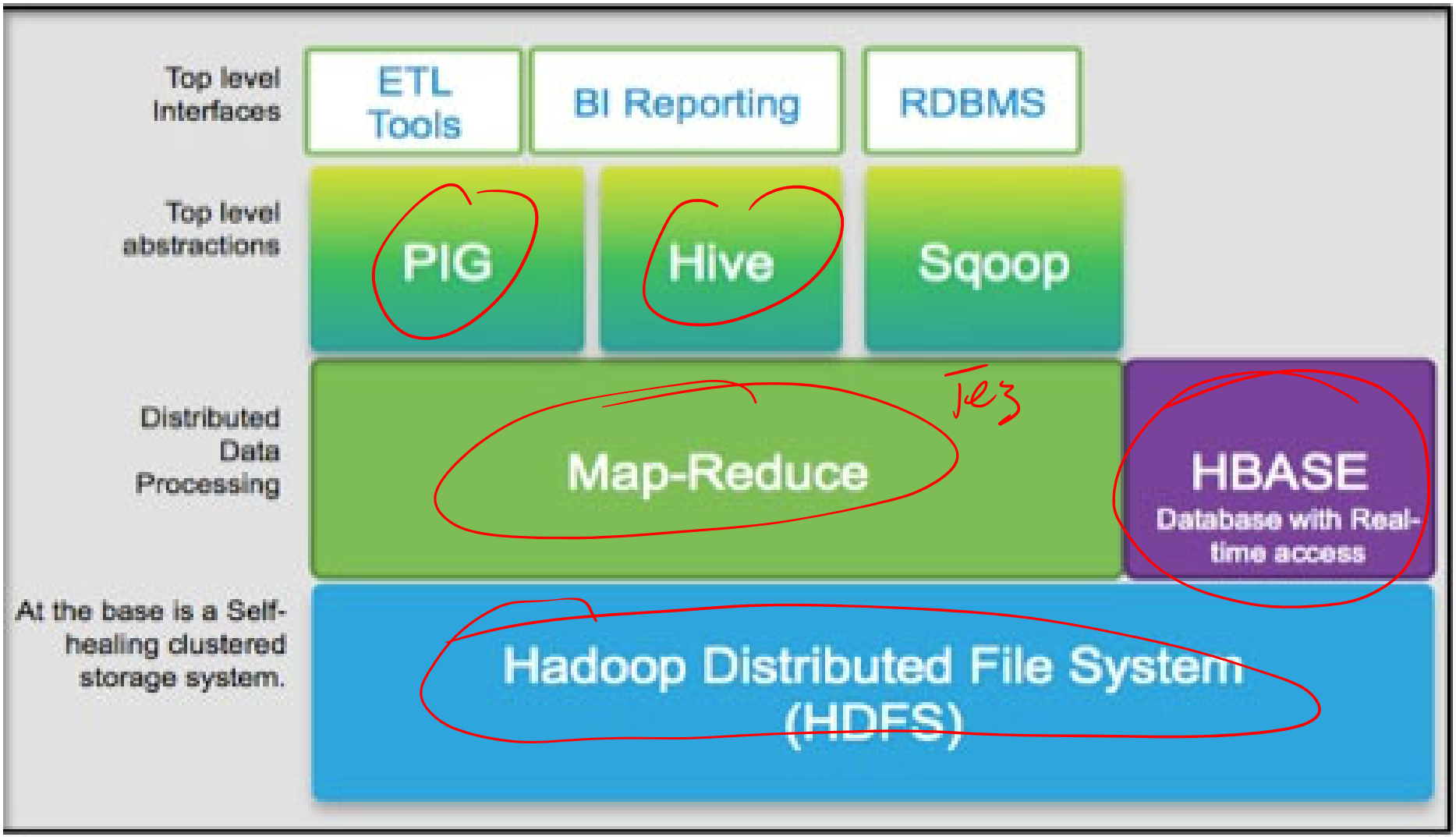


Table1
row keys
cd_fam1
a b c d
fam2
x y
fam4
c1 c2 c3 c4
s
1
2
3
4
5
6
7
8
9
10
11
12
14
1
7
a
b
1 z
1 1
x
a
b
d
1
2
4
5

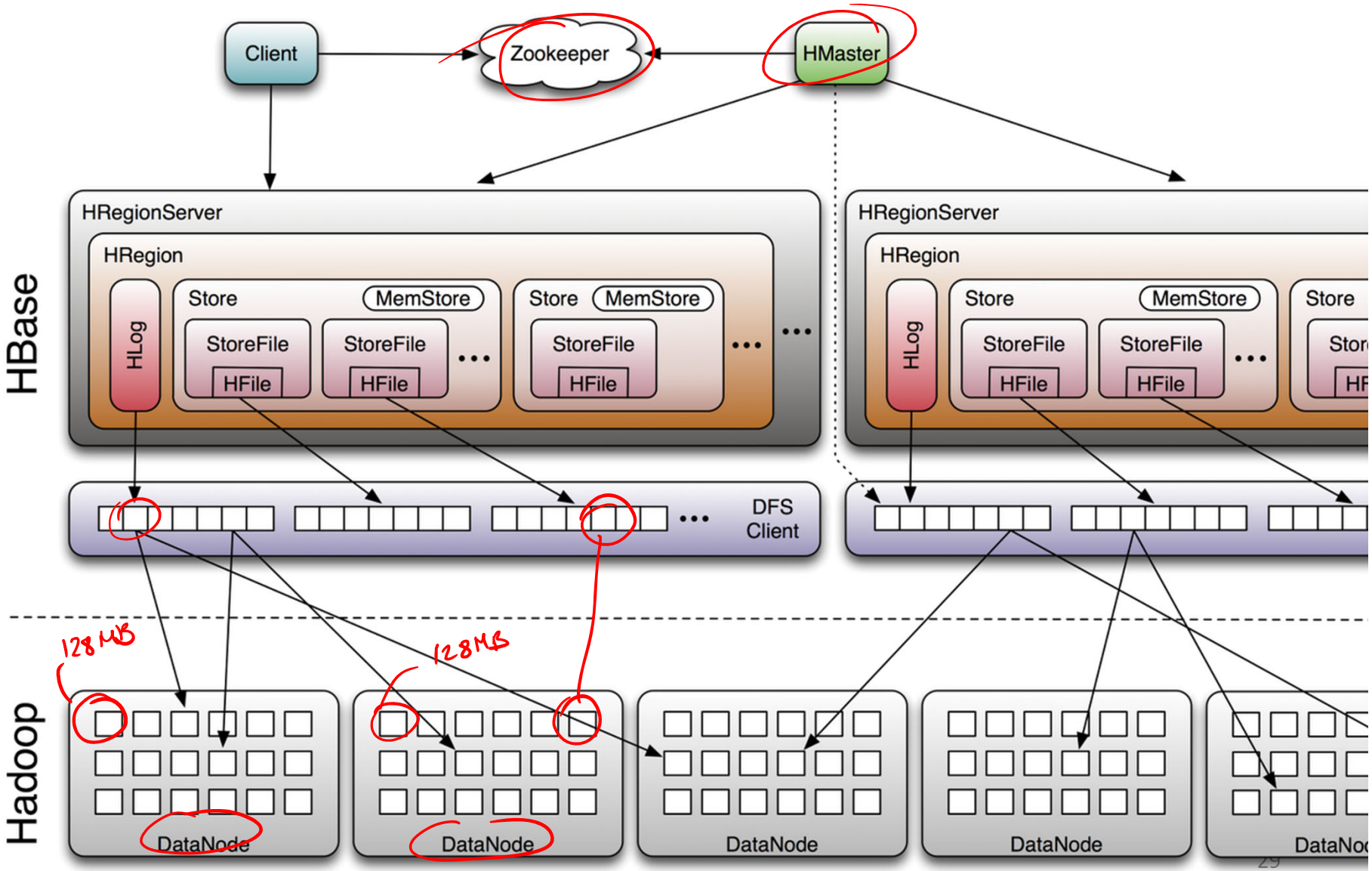
HBase

- Column-oriented store
  - Google's Bigtable
- Random, real-time updates
  - Individual record access
- Highly available/distributed
- No SQL
  - No joins
  - (some) indexing

HBase

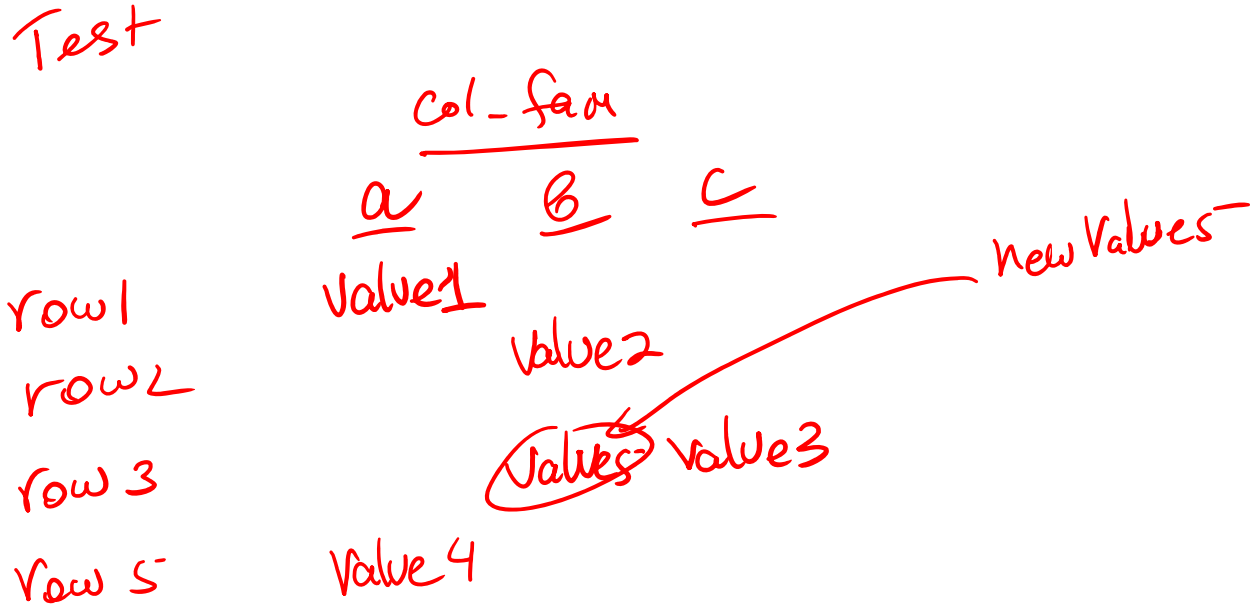

Top level Interfaces
ETL Tools
BI Reporting
RDBMS
Top level abstractions
PIG
Hive
Sqoop
Distributed Data Processing
Map-Reduce
HBASE
Database with Real-time access
At the base is a Self-healing clustered storage system.
Hadoop Distributed File System (HDFS)

HBase Architecture



HBase Example

- Create/populate a table:
  - create 'test', 'col_fam'
  - list 'test'
  - put 'test', 'row1', 'col_fam:a', 'value1'
  - put 'test', 'row2', 'col_fam:b', 'value2'
  - put 'test', 'row3', 'col_fam:c', 'value3'
  - put 'test', 'row5', 'col_fam:a', 'value4'
  - put 'test', 'row3', 'col_fam:b', 'value5'
  - scan 'test'


Test
col_fam
a
B
C
row1
row2
row3
row5
value1
value2
value3
value4
values
new values

Using HBase

```
put 'test', 'row1', 'col_fam:z', 'value6'
put 'test', 'row5', 'col_fam:z', 'value7'
put 'test', 'row6', 'col_fam:z', 'value8'
scan 'test', {COLUMNS => 'col_fam:c'}
scan 'test', {STARTROW => 'row3'}
put 'test', 'row3', 'col_fam:b', 'newValue5'

scan 'test', {COLUMNS => 'col_fam:b'}
scan 'test', {TIMERANGE=>[a, b]}
```

We Stopped Here

Hive SELECT TRANSFORM

```
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;
```

SQL to MapReduce

```
SELECT AVG(Grade) as GPA, FirstName,
        COUNT(DISTINCT LastName), MAX(s.Year)
FROM Student s, Grades g
WHERE s.ID = g.SID
GROUP BY FirstName
ORDER BY GPA
```

Next Time:

- Performance and Compression
- NoSQL
- DNS and Firewalls
- Multi-node cluster setup