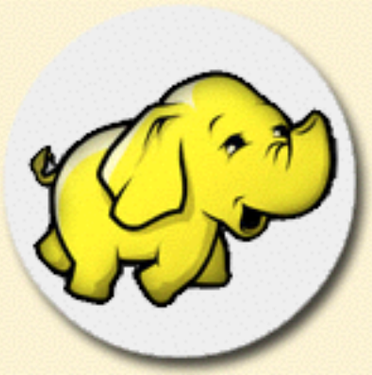


Welcome to

Big Data & Hadoop

Session

Session 2 - Environment Overview, MapReduce Basics

+91-9538998962

sandeep@knowbigdata.com

WELCOME - KNOWBIGDATA

- Interact - Ask Questions
- Lifetime access of content
- Class Recording
- Cluster Access
- 24x7 support
- Real Life Project
- Quizzes & Certification Test
- 11 x (3hr class)
- Socio-Pro Visibility
- Mock Interviews

ABOUT ME

2014	**KnowBigData**	Founded
2014	**Amazon**	Built High Throughput Systems for Amazon.com site using in-house NoSql.
2012		
2012	**InMobi**	Built Recommender after churning 200 TB
2011		
	tBits Global	Founded tBits Global Built an enterprise grade Document Management System
2006		
	D.E.Shaw	Built the big data systems before the term was coined
2002		
2002	**IIT Roorkee**	Finished B.Tech somehow.

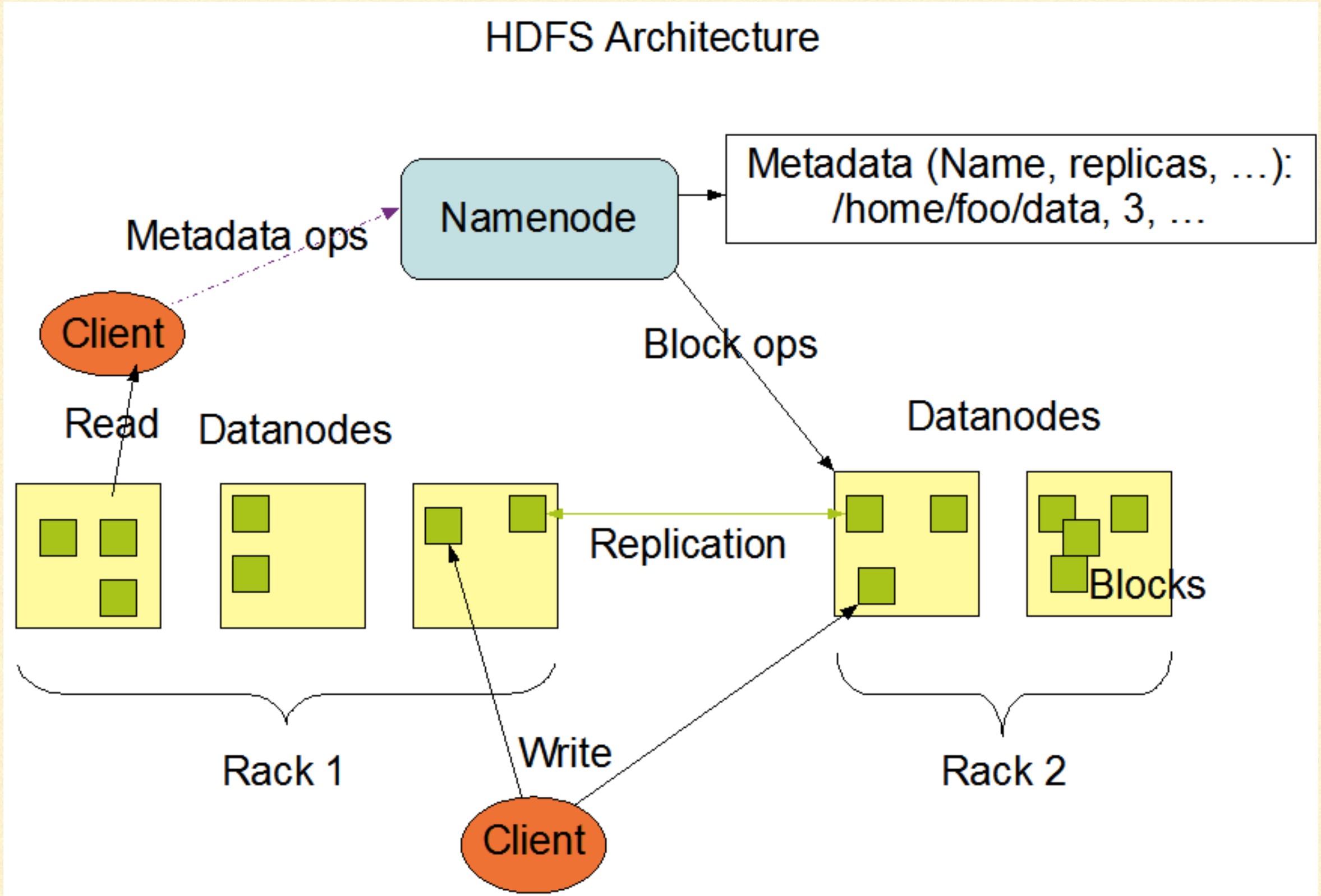
COURSE CONTENT

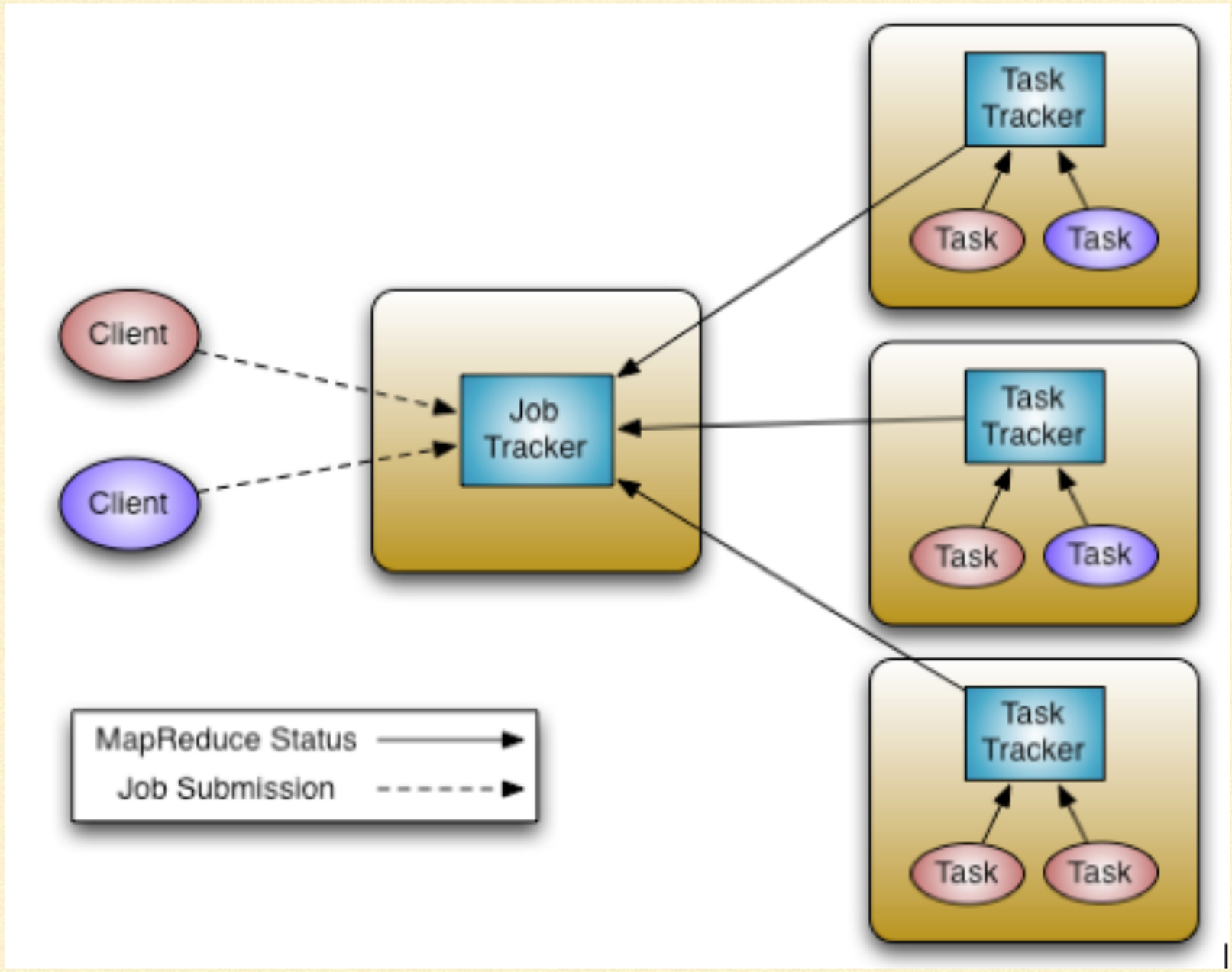
I	Understanding BigData, Hadoop Architecture
II	Environment Overview, MapReduce Basics
III	MapReduce framework
IV	Adv MapReduce & Testing
V	Analytics using Pig
VI	Analytics using Hive
VII	NoSQL, HBASE
VIII	Zookeeper, Oozie, Mahout
IX	Apache Flume, Apache Spark
X	YARN, Big Data Sets & Project Assignment

TODAY'S CLASS

- Recap
  - HDFC Architecture
  - Hadoop 1.0 Architecture
  - Hadoop 2.0 / Yarn
- Cluster Overview
  - Using Cloudera Manager
  - Hue
- Using HDFS, Hive, Pig, Oozie
  - From Web
  - From Command
- Thinking in Map/Reduce
  - Frequency Problem
    - Solution 1 - Coding
    - Solution 2 - SQL
    - Solution 3 - Unix Pipes
    - Solution 4 - External Sort
  - Map/Reduce Overview
  - Visualisation
  - Analogies to groupby
  - Assignments

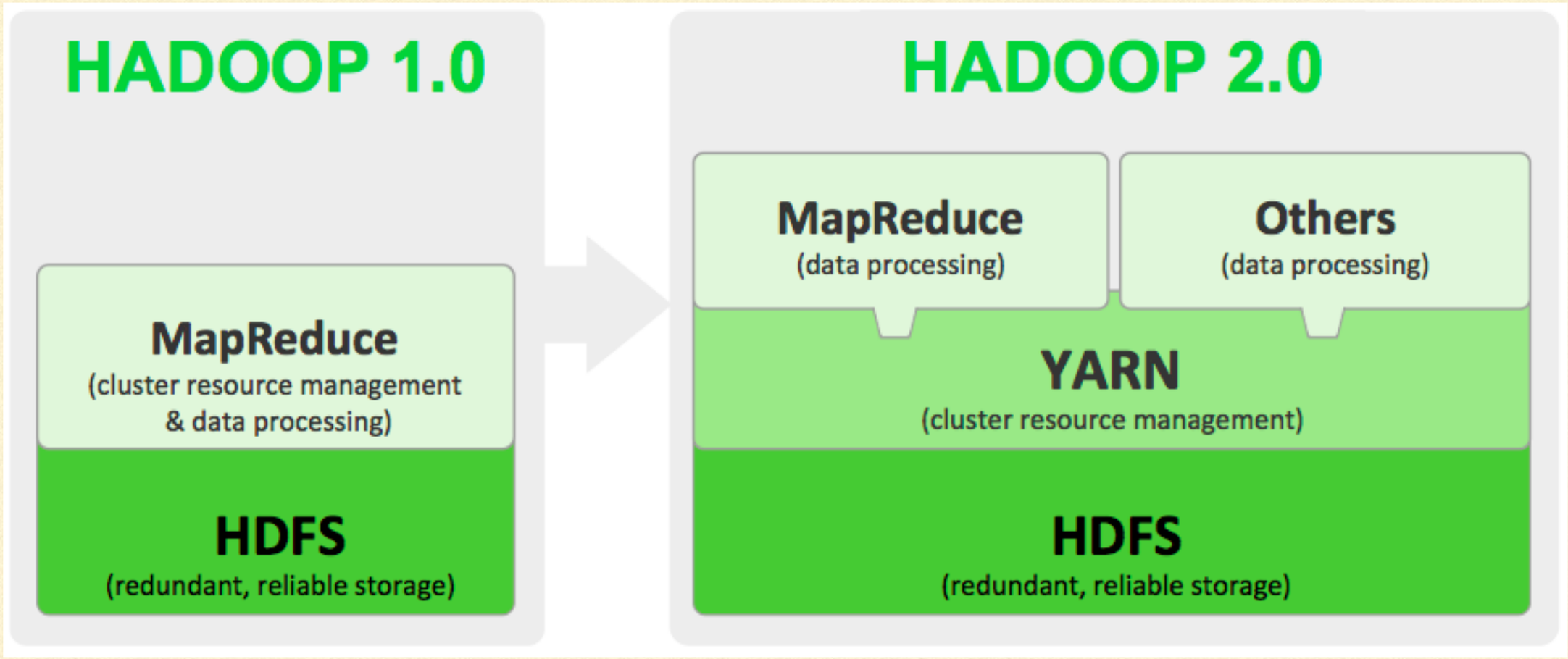
RECAP


HDFS Architecture
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Client
Rack 2


Client
Client
Job Tracker
Task Tracker
Task
Task
Task Tracker
Task
Task
Task Tracker
Task
Task
MapReduce Status
Job Submission

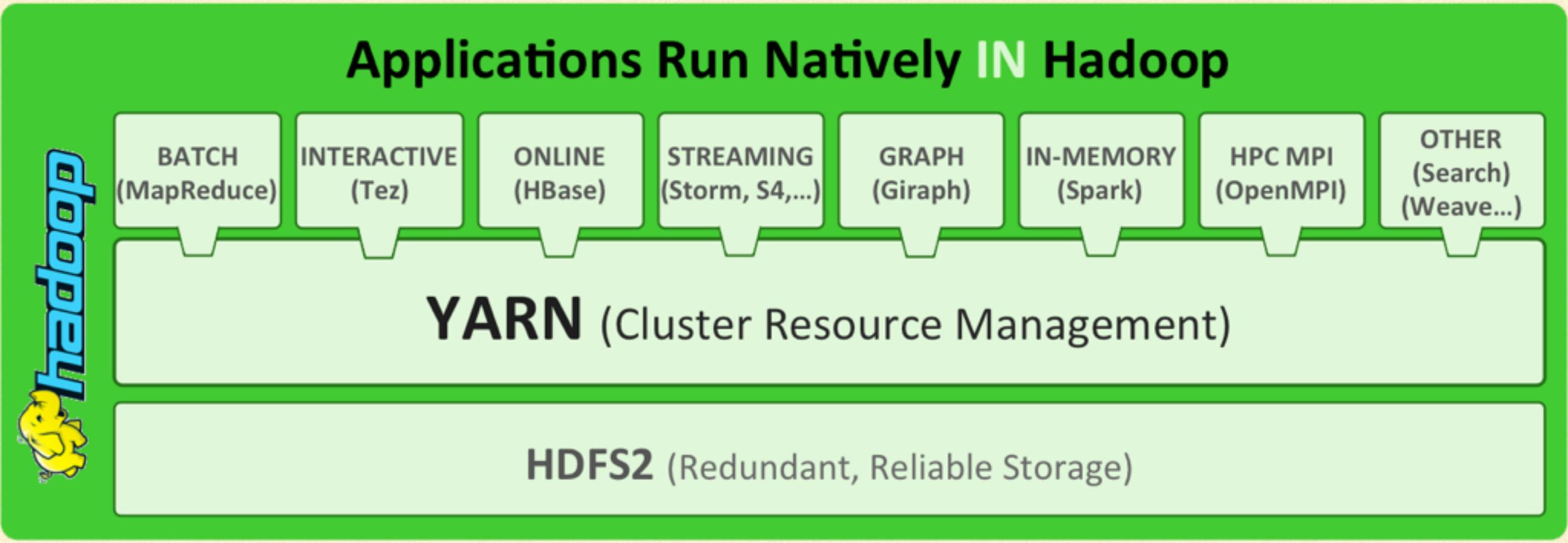
YARN

YET ANOTHER RESOURCE NEGOTIATOR



YARN

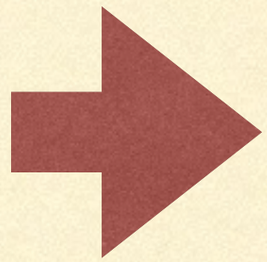
YET ANOTHER RESOURCE NEGOTIATOR



YARN

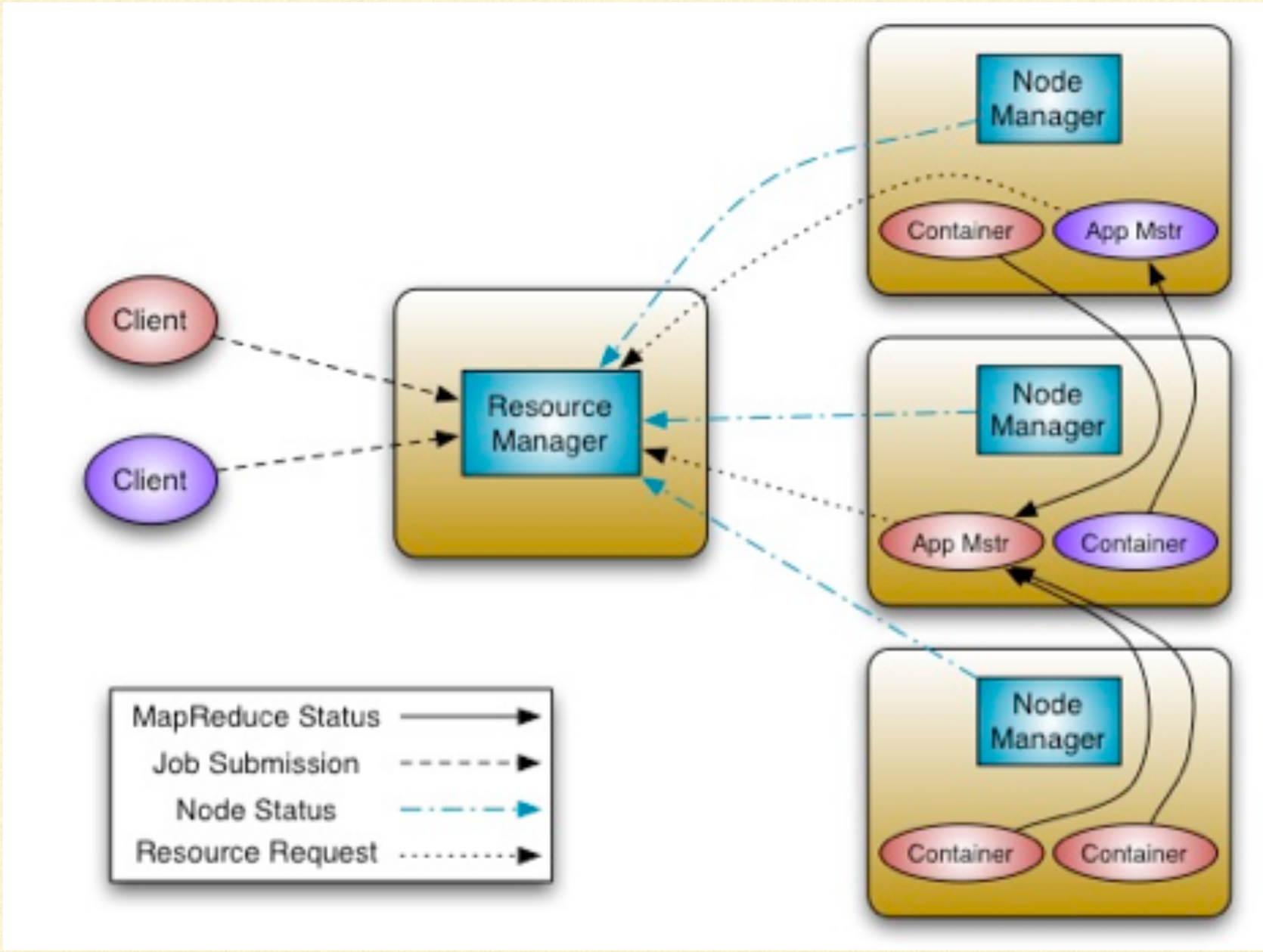
- Support for workloads other than MapReduce
- Scalability
- Compatibility with MapReduce
- Improved cluster utilization.
- Agility
- Map Reduce was batch oriented

JobTracker
TaskTracker



Global ResourceManager
Per-application ApplicationMaster.
Per-node slave NodeManager and
Per-application Container running on a NodeManager

YARN
YET ANOTHER RESOURCE NEGOTIATOR



QUICK - CLUSTER HANDS ON

- Seeing the status: Web Interface
- SSH Connecting to Cluster
- Mkdir & Copy on HDFS
- MapReduce Command
- Pig Command
- Hive Command

QUICK - CLUSTER HANDS ON

Seeing the status

Name Node:
hadoop1.knowbigdata.com

Data Node:
hadoop[2,3,4].knowbigdata.com

Ambari - Manages the servers
http://hadoop1.knowbigdata.com:8080

Hue - interact with services
http://hadoop1.knowbigdata.com:8000

QUICK - CLUSTER HANDS ON

SSH Connection <demo>

On Windows:

Use, putty to ssh

Use WinSCP for copying files

On Unix/Mac:

To Login:

ssh student@hadoop1.knowbigdata.com

To Copy Files:

scp mylocalfile student@hadoop1.knowbigdata.com:

QUICK - CLUSTER HANDS ON

Mkdir & Copy on HDFS

Using web interface

Using Command Line

hadoop fs -moveFromLocal big.txt /user/student/wordcount/input/

Please create your own folder and put your work there

QUICK - CLUSTER HANDS ON

Pig + hive Command

Pig:

```
pig
A = load '/user/student/wordcount/output/part*'
as (f1:chararray, f2:int);
dump A;
```

hive

```
hadoop fs -cat /user/student/wordcount/output/part* > freq
CREATE TABLE pokes (bar STRING,foo int);
LOAD DATA LOCAL INPATH './freq' OVERWRITE INTO TABLE
pokes;
select * from pokes;
```

QUICK - CLUSTER HANDS ON

MapReduce Command

The Example is available here

Remove old output directory

hadoop fs -rm -r /user/student/wordcount/output

Upload the MapReduce Jar

hadoop fs -cp mapred.jar /root/giri/mapred.jar

Execute the mapReduce Command:

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount /user/student/sgiri/wordcount/input /user/student/sgiri/wordcount/output4

*hadoop fs -cat /user/student/wordcount/output/part**

THINKING IN MAP / REDUCE

If you have the plain text file of all the Lord Of The Rings books, [10 gb] how would you find the frequencies of words?

THINKING IN MAP / REDUCE

If you have the plain text file of all the Lord Of Rings books, how would you find the frequencies of words?

Approach 1 (Programmatic):

- Create a frequency hash table
- For each word in the file
- Increase its frequency in the hash table
- When no more words left in file, print the hash table

Problems?

THINKING IN MAP / REDUCE

If you have the plain text file of all the Lord Of Rings books, how would you find the frequencies of words?

Approach 1 (Programmatic):

- Create a hash table
- For each word in the file
- Increase its frequency in the hash table
- When no more words left in file, print the hash table

Problems?
Can not process the data beyond RAM size.

THINKING IN MAP / REDUCE

If you have the plain text file of all the Lord Of Rings books, how would you find the frequencies of words?

Approach2 (SQL):

- Break the books into one word per line
- Insert one word per row in database table
- Execute: *select word, count(*) from table group by word.*

THINKING IN MAP / REDUCE

If you have the plain text file of all the Lord Of Rings books, how would you find the frequencies of words?

Approach 3 (Unix):

- Replace space with a newline
- Order lines with a sort command
- Then find frequencies using uniq
  - Scans from top to bottom
  - prints the count when line value changes

cat file1 file2 … | sed 's/[\t]+/\n/g' | sort -S 1g| uniq -c

THINKING IN MAP / REDUCE

Problems in Approach 2 (SQL) & Approach 3 (Unix)?

Problems in Approach 2 (SQL) & Approach 3 (Unix)?

The moment the data starts going beyond RAM the time taken starts increasing. The following become bottlenecks:

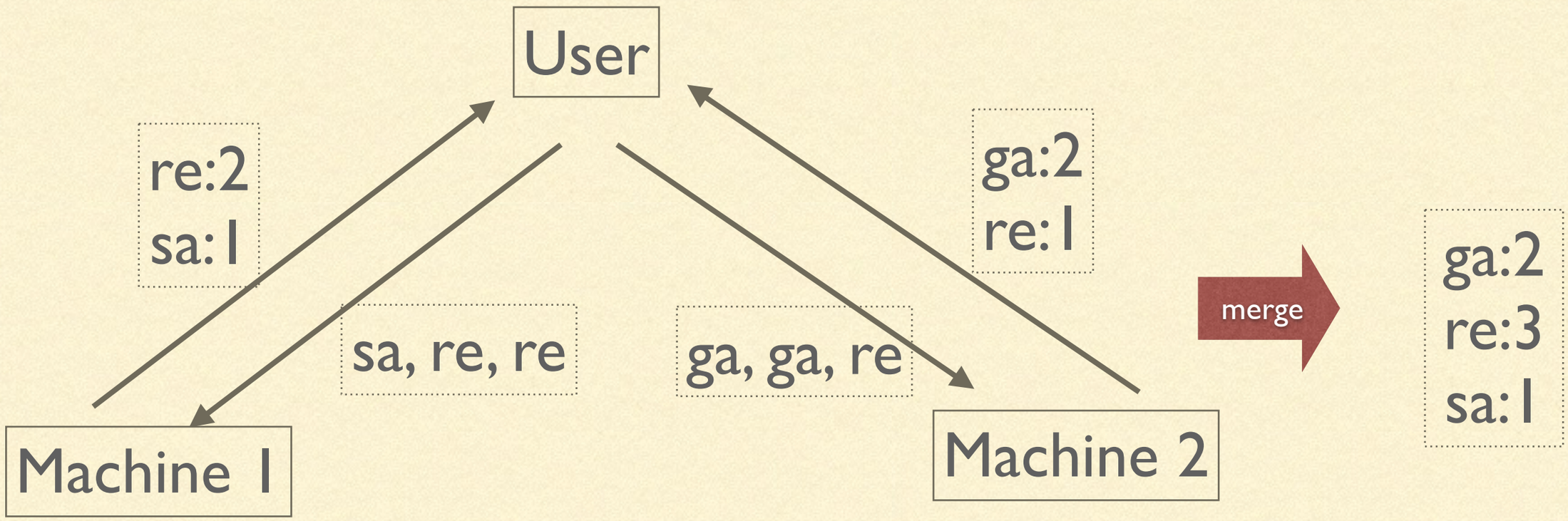
- RAM
- Network Speed
- Disk Space
- Disk Speed
- CPU

THINKING IN MAP / REDUCE

Then?

Use a external sort.

- Split the files to a size that fits RAM
- Use the previous approaches (2&3) to find freq
- Merge (sort -m) and sum-up frequencies



THINKING IN MAP / REDUCE

Problems with external Sort?

THINKING IN MAP / REDUCE

Problems with external Sort?

Time is consumed in transport of data.
+
For each requirement we would need to special purpose network oriented program.
/
Would Require A lot of Engineering.

Solution?
Use Map/Reduce

THINKING IN MAP / REDUCE

What is Map/Reduce?

- Programming Paradigm
  - To help solve Big Data problems
  - Specifically sorting intensive jobs
- You would have to code two functions:
  - Mapper - Convert Input into “key - value” pairs
  - Reducer - Aggregates all the values for a key
- Also supported by many other systems such as
  - MongoDB
  - CouchDB
- Mapper & Reducers
  - can be written in Java, Shell, Python or any binary

MAP / REDUCE

Mapper/Reducer for word frequency problem.

sa re sa ga

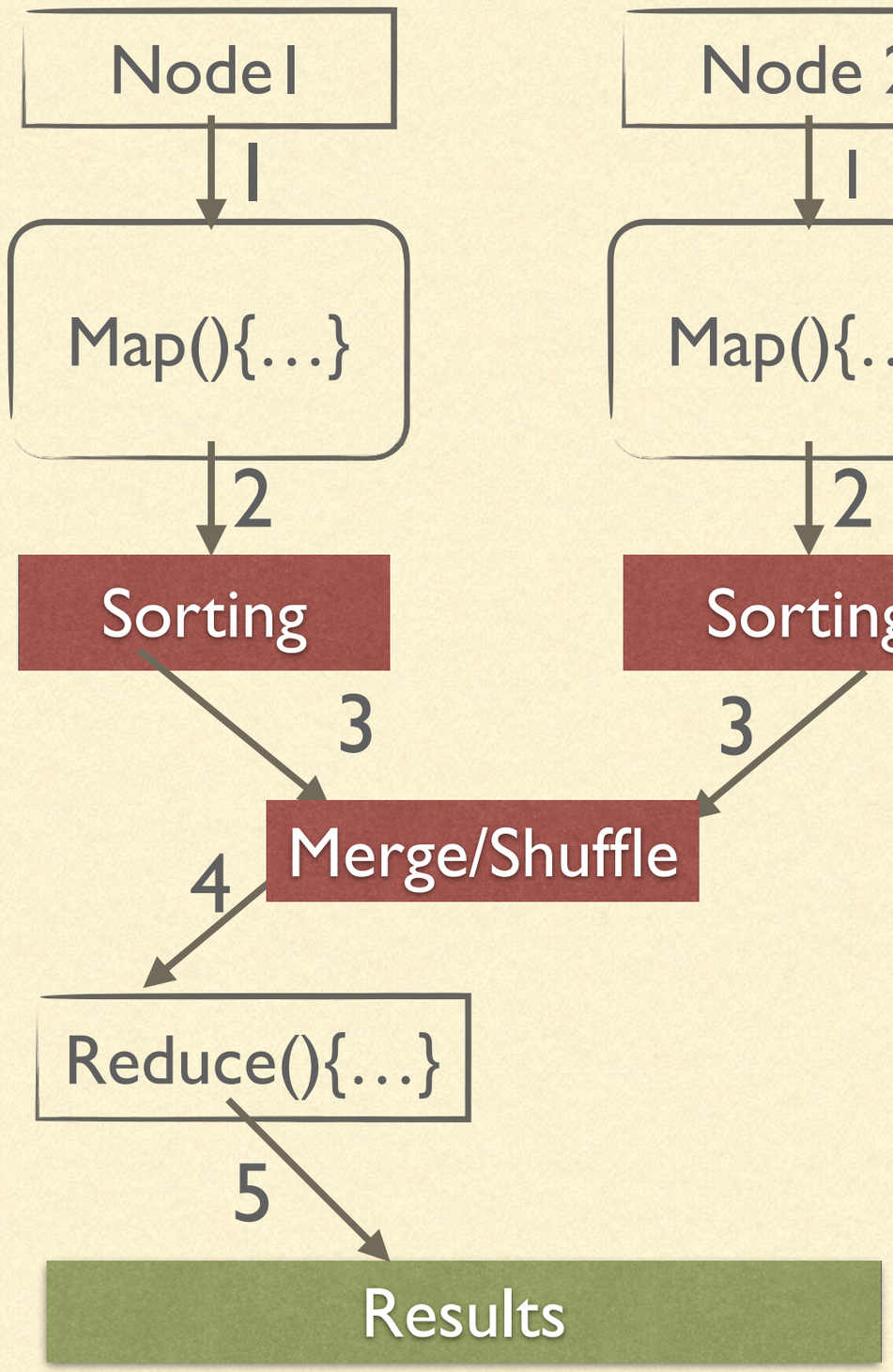
```
function map(input):
    foreach(word in input) :
        print(word, 1);
```

sa 1
re 1
sa 1
ga 1

ga [1]
re [1]
sa [1, 1]

```
function reduce(word, freqArray):
    return Array.sum(freqArray);
```

ga 1
re 1
sa 2

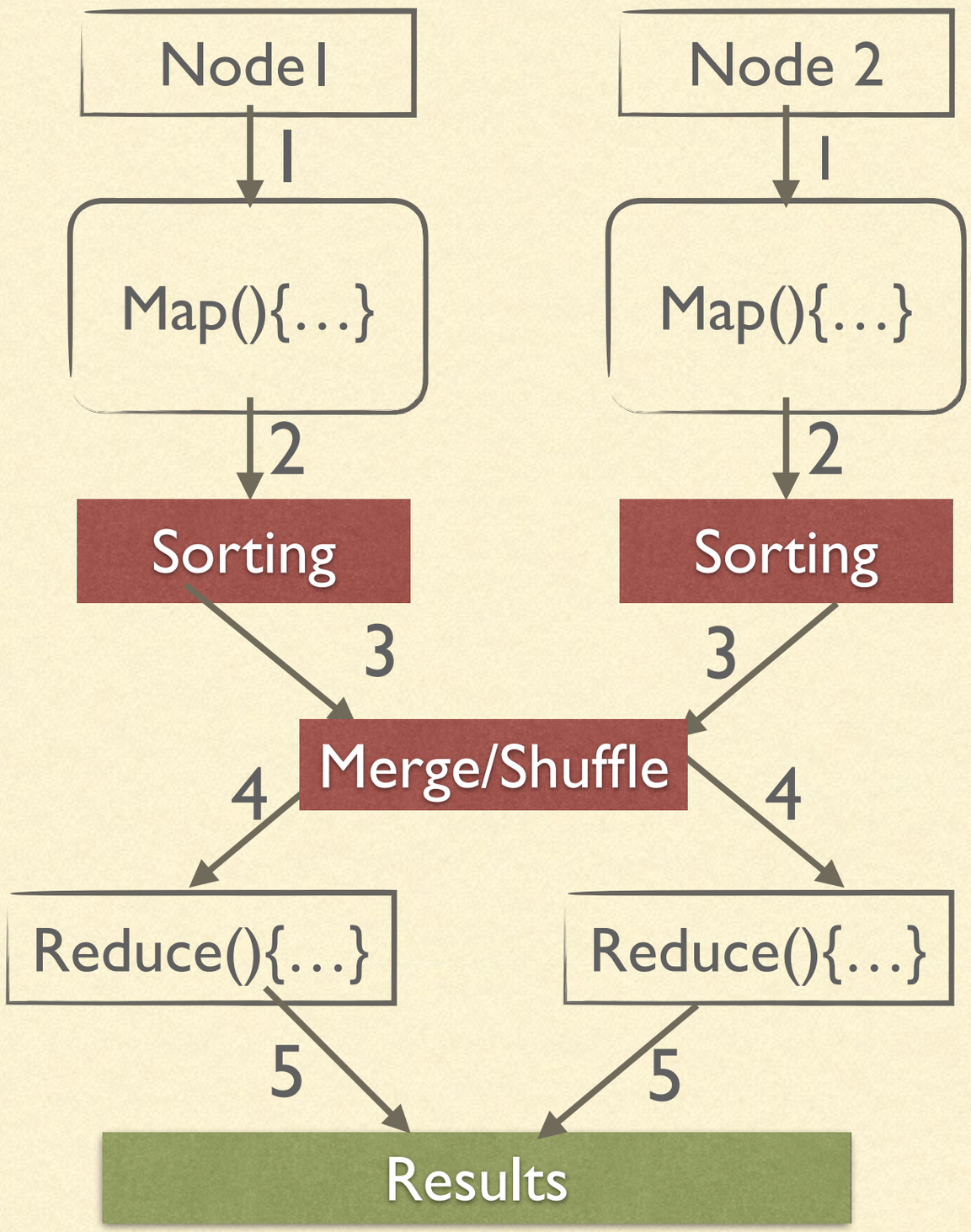


ga:1, re:2, sa:4

1: sa re sa ga
2.
sa 1
re 1
sa 1
ga 1
3.
ga 1
re 1
sa 1
sa 1
4.
ga 1
re [1,1]
sa [1,1,1,1]

1: sa re sa
2.
sa 1
re 1
sa 1
3.
re 1
sa 1
sa 1
5.
ga 1
re 2
sa 4

"sa re sa ga"



ga:1, re:2, sa:4

"sa re sa"

1: sa re sa ga
2.
sa 1
re 1
sa 1
ga 1
3.
ga 1
re 1
sa 1
sa 1
4.
ga 1
re 1
re 1
5.
ga 1
re 2

1: sa re sa
2.
sa 1
re 1
sa 1
3.
re 1
sa 1
sa 1
4.
sa 1
sa 1
sa 1
sa 1
5.
sa 4

MAP / REDUCE

Analogous to Group By

select word, count(*) from table group by word.

```
function map():
    foreach(word in input) :
        print(word, 1);
```

```
function reduce(word, freqArray):
    return Array.sum(freqArray);
```

Assignment For Tomorrow

1. Frequencies of letters [a-z] - Do you need Map/Reduce?

2. Find anagrams in a huge text. An anagram is basically a different arrangement of letters in a word.

Input:

"the cat act in tic tac toe."

Output:

cat, tac, act

the

toe

in

Assignment For Tomorrow

3a. A file contains the DNA sequence of people. Find all the people who have same DNAs.

Input:

"User1 ACGT"
"User2 TGCA"
"User3 ACG"
"User4 ACGT"
"User5 ACG"
"User6 AGCT"

Output:

User1, User4
User2
User3, User 5
User6

Assignment For Tomorrow

3b. A file contains the DNA sequence of people. Find all the people who have same or mirror image of DNAs.

Input:

"User1 ACGT"
"User2 TGCA"
"User3 ACG"
"User4 ACGT"
"User5 ACG"

Output:

User1, User2, User4
User3, User 5

Assignment For Tomorrow

4. In an unusual democracy, everyone is not equal. The vote count is a function of worth of the voter. Though everyone is voting for each other.
As example, if A with a worth of 5 and B with a worth of 1 are voting for C, the vote count of C would be 6.
You are given a list of people with their value of vote. You are also given another list describing who voted for who all.

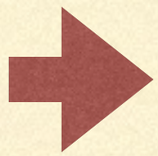
Find out what is the vote count of everyone?

List1

Voter	*Votee*
A	C
B	C
C	F

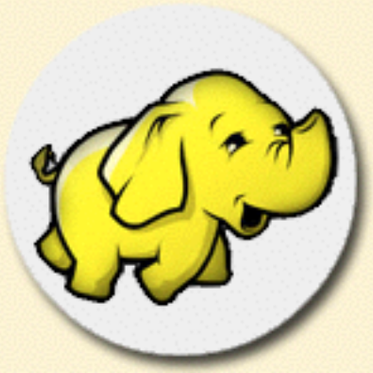
List2

Person	*Worth*
A	5
B	1
C	11



Result

Person	*VoteCount*
A	0
B	0
C	6
F	11



Big Data & Hadoop

Thank you.

+91-9538998962

sandeep@knowbigdata.com