CS15-319 / 15-619
Cloud Computing

Recitation 3

January 28th & 30th, 2014

Bugs!

- Question 3 in Checkpoint 1
  - We have manually graded it

Amazon Web Services (AWS) Account

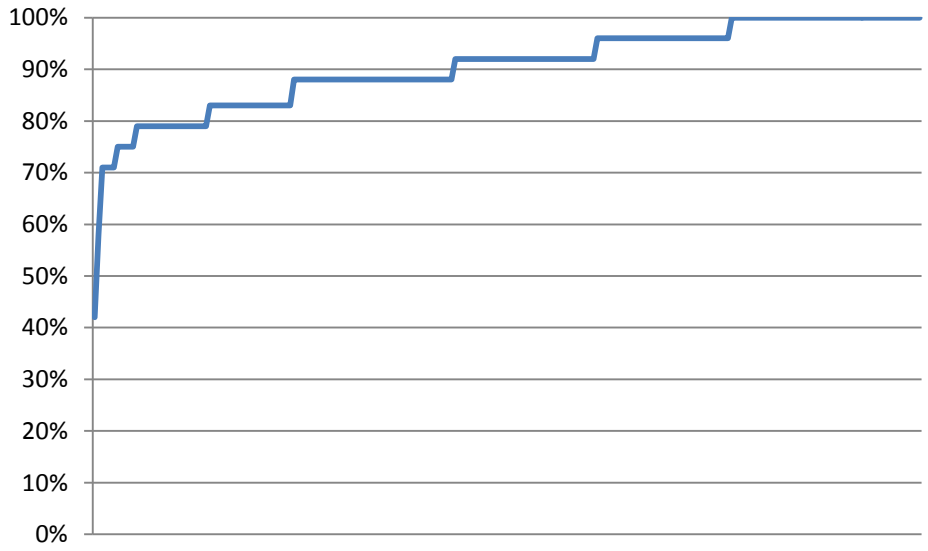
- For students who just enrolled:
  - Create your AWS account
  - Fill the Google form with your AWS account info
  - Send your account information to Jason Boles (jboles@cmu) with Email Subject: Request to add account to Consolidated Bill

Last Week

- Unit1: Introduction to Cloud Computing
  - Introduction, Building Blocks, Service Models
  - Checkpoint Quiz 1
- Project 1: Introduction to Big Data Analysis
  - Wikimedia data set of Wikipedia page views
  - Parse and filter the data
  - Sort the data and save the output to a file

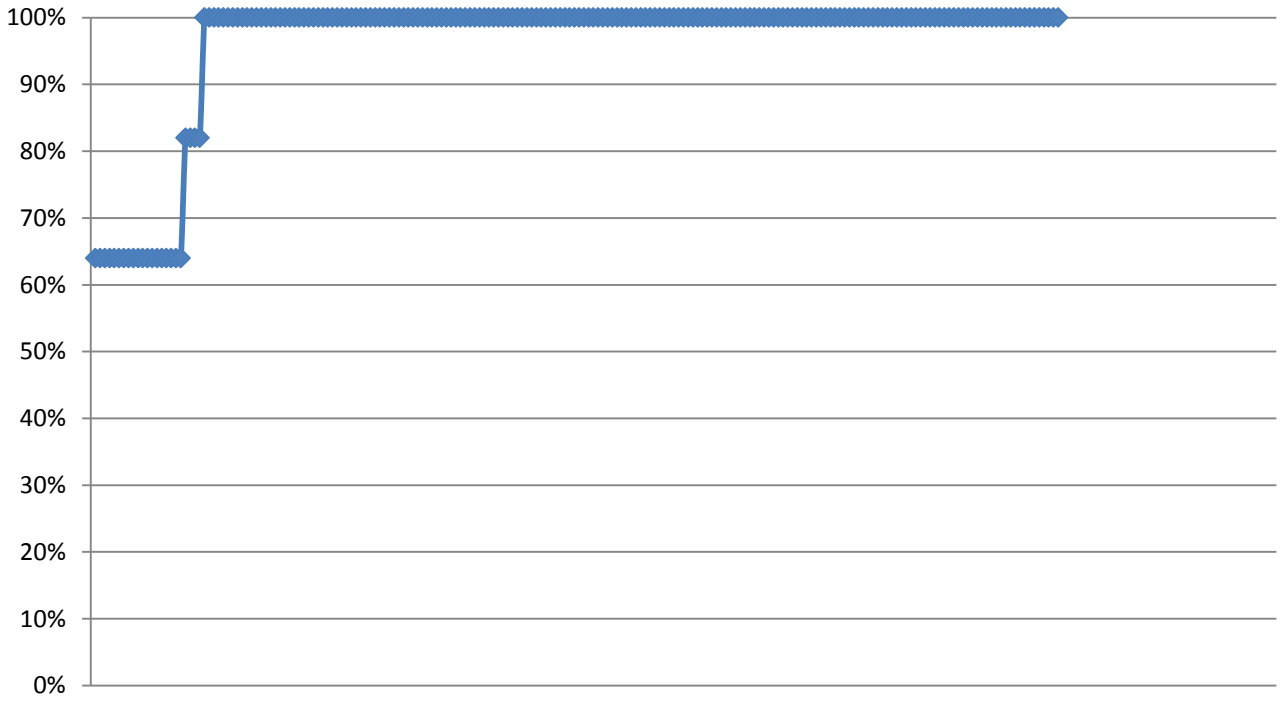
Quiz 1

- Stats
  - Average is 90%
  - Grades:



Project 1 Checkpoint 1

- Introduction to Big Data:
  - Sequential Analysis: Average is: 96%
  - Elastic MapReduce: this week



Student Questions on Piazza

- Cannot find AMI
  - Choose the correct region:
    - us-east-1 (Northern Virginia)
  - If still not found, wait for some time.

Please ask public questions when possible

Unit 2: Data Centers

- Start reading first 2 modules in Unit 2:
  - Module 3: Data Center Trends
  - Module 4: Data Center Components
  - Module 5: Design Considerations
  - Unit 2: Checkpoint Quiz

UNIT 2: Data Centers			
Module 3: Data Center Trends Module 4: Data Center Components Module 5: Design Considerations			
Quiz 2: Data Centers		Checkpoint	Not yet assigned Due date TBD by instructor

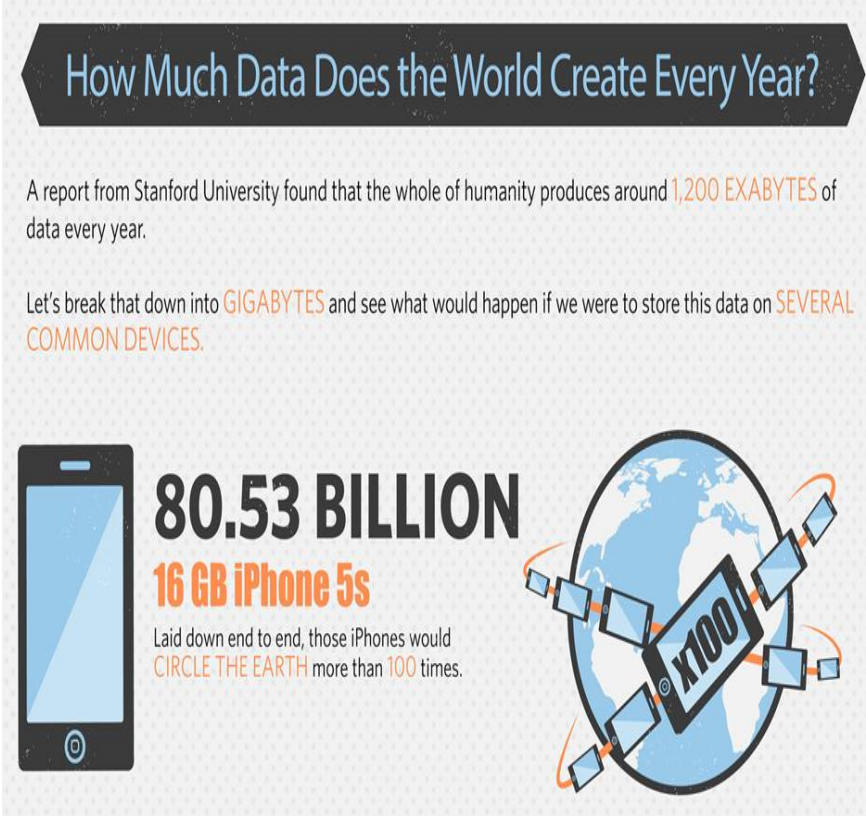
Motivation for MapReduce

- The problem

How many times does each term appear in all books in Hunt Library?

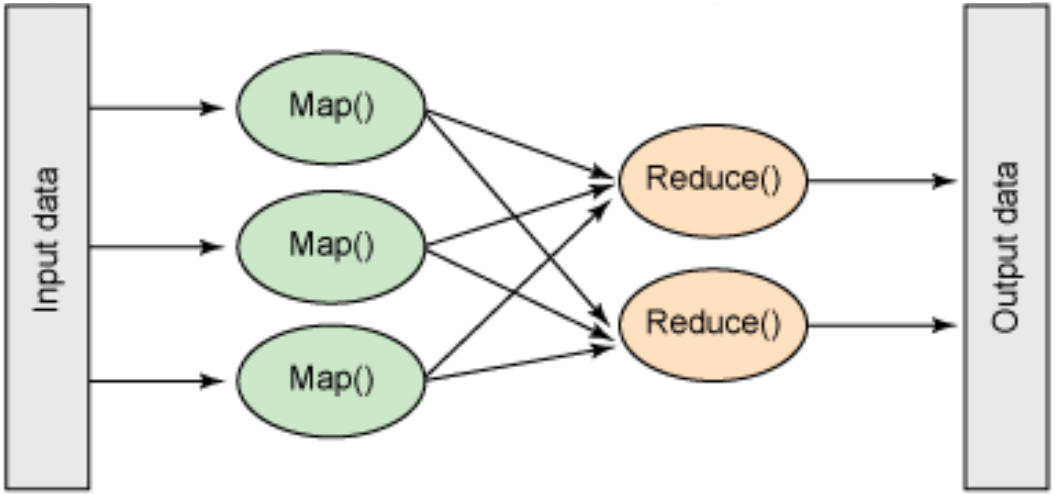
Motivation for MapReduce

- When the file size is 200MB
  - HashMap: (term, count)
  - Doable on a single machine
- When the file size is 200TB
  - Out of memory
  - Slow
  - How would you deal with it?
    - Partition the input?
    - Distribute the work?
    - Coordinate the effort?
    - Aggregate the results?



Definition of MapReduce

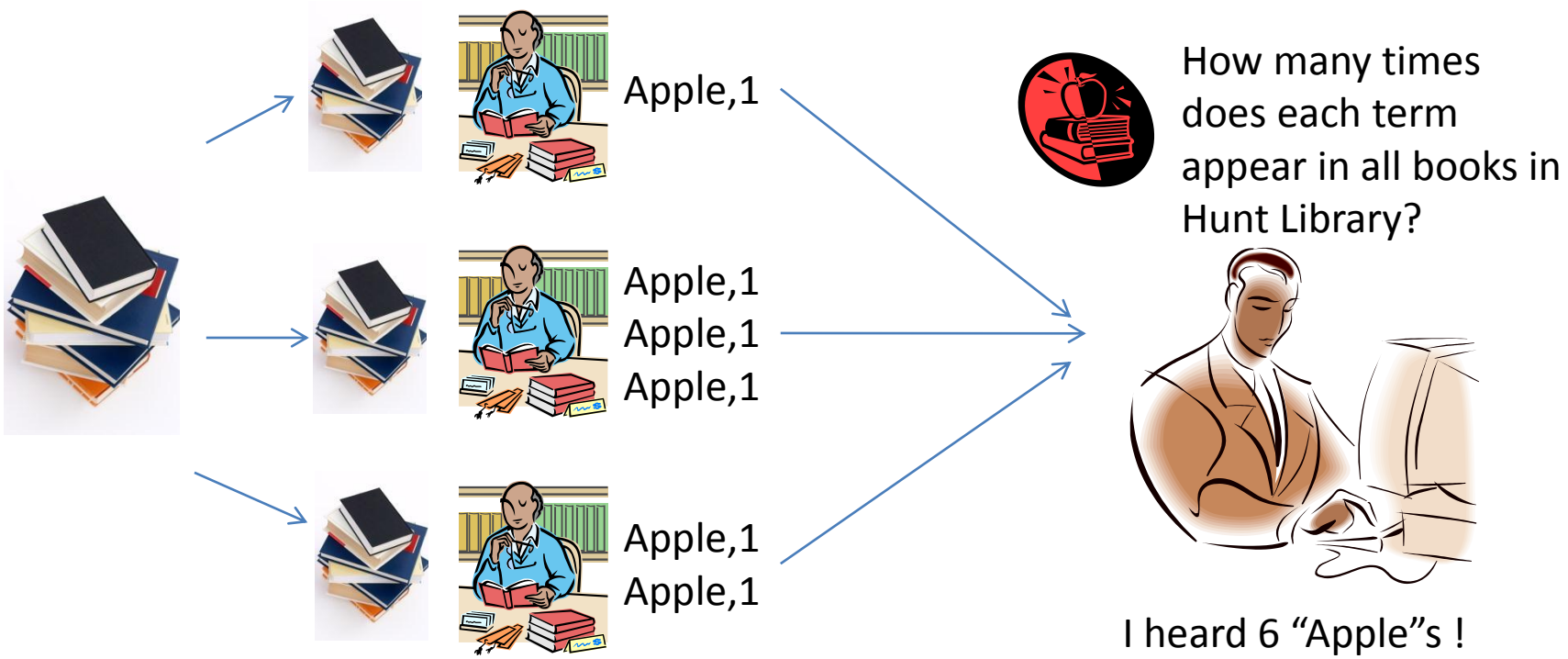
- Programming model for processing large data sets with a parallel, distributed algorithm on a cluster



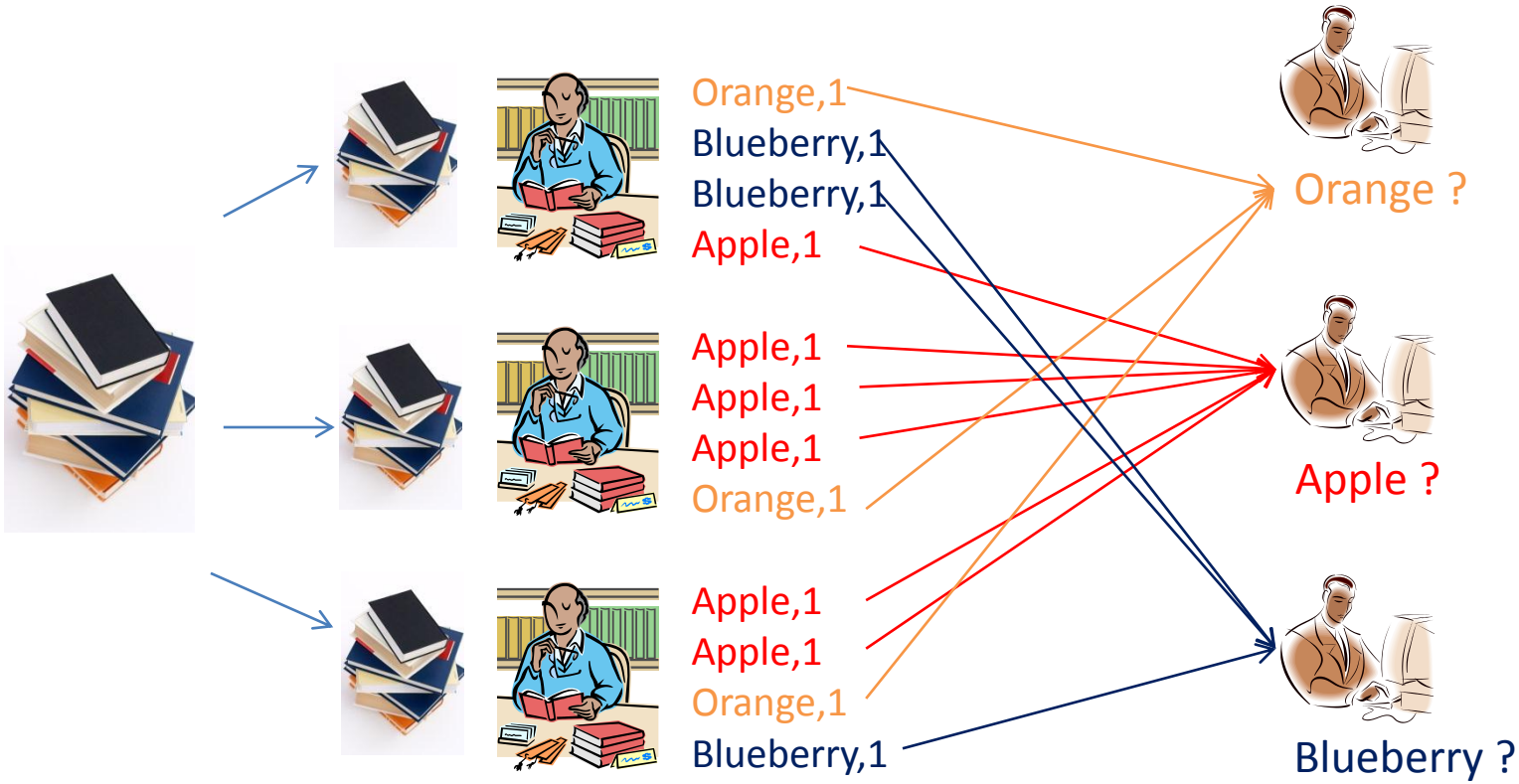
MapReduce in 15-319/619

- In this course we are going to use MapReduce on 2 Platforms:
  1. Amazon Elastic MapReduce (this week)
  2. Hadoop (later projects)

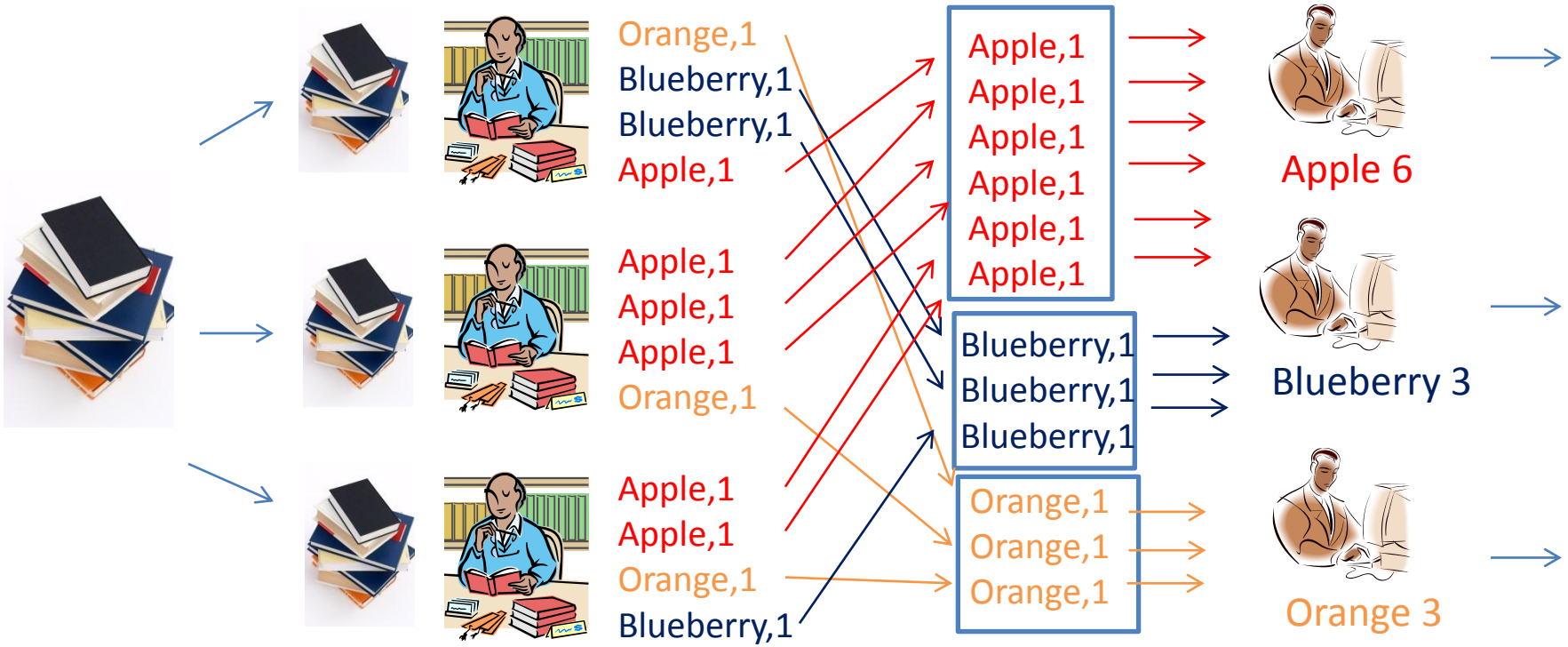
MapReduce Example



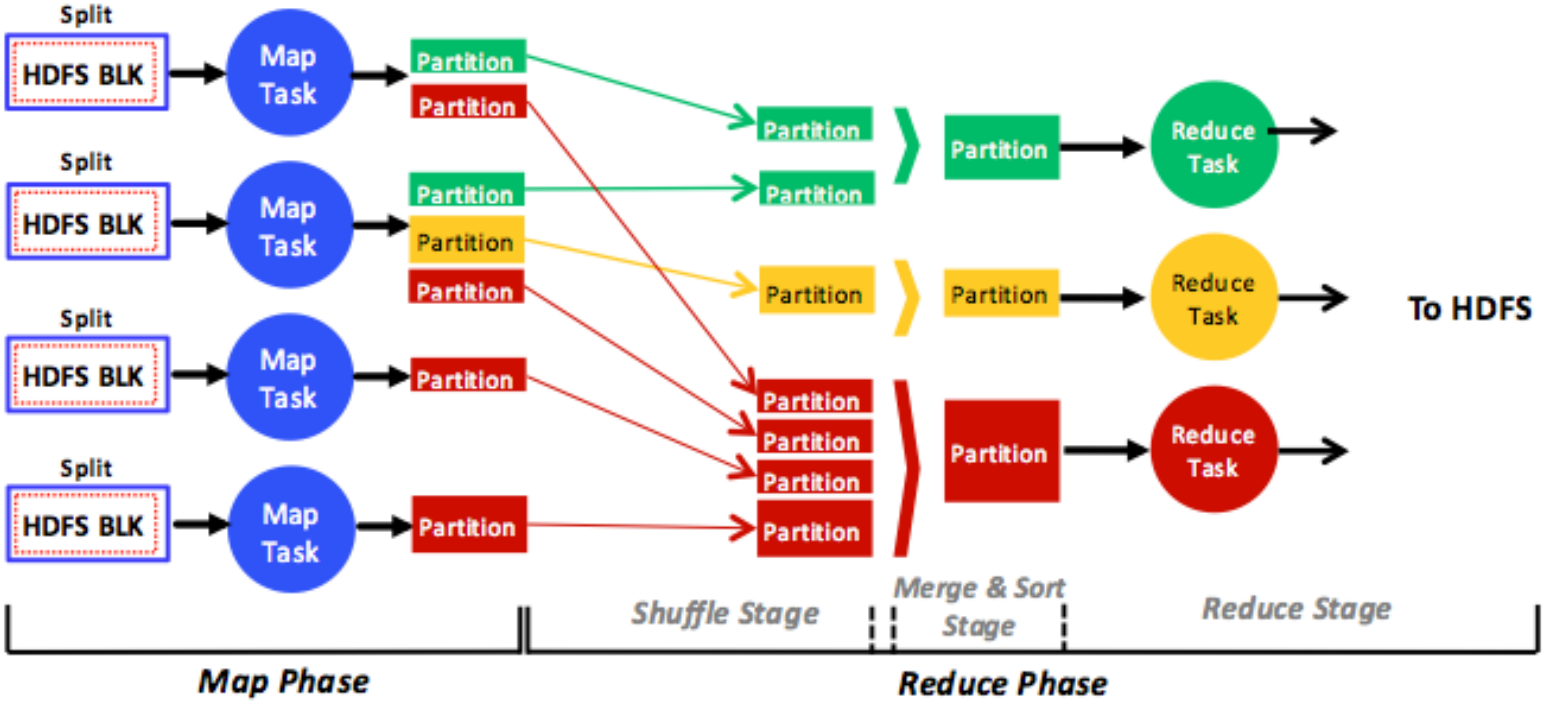
MapReduce Example



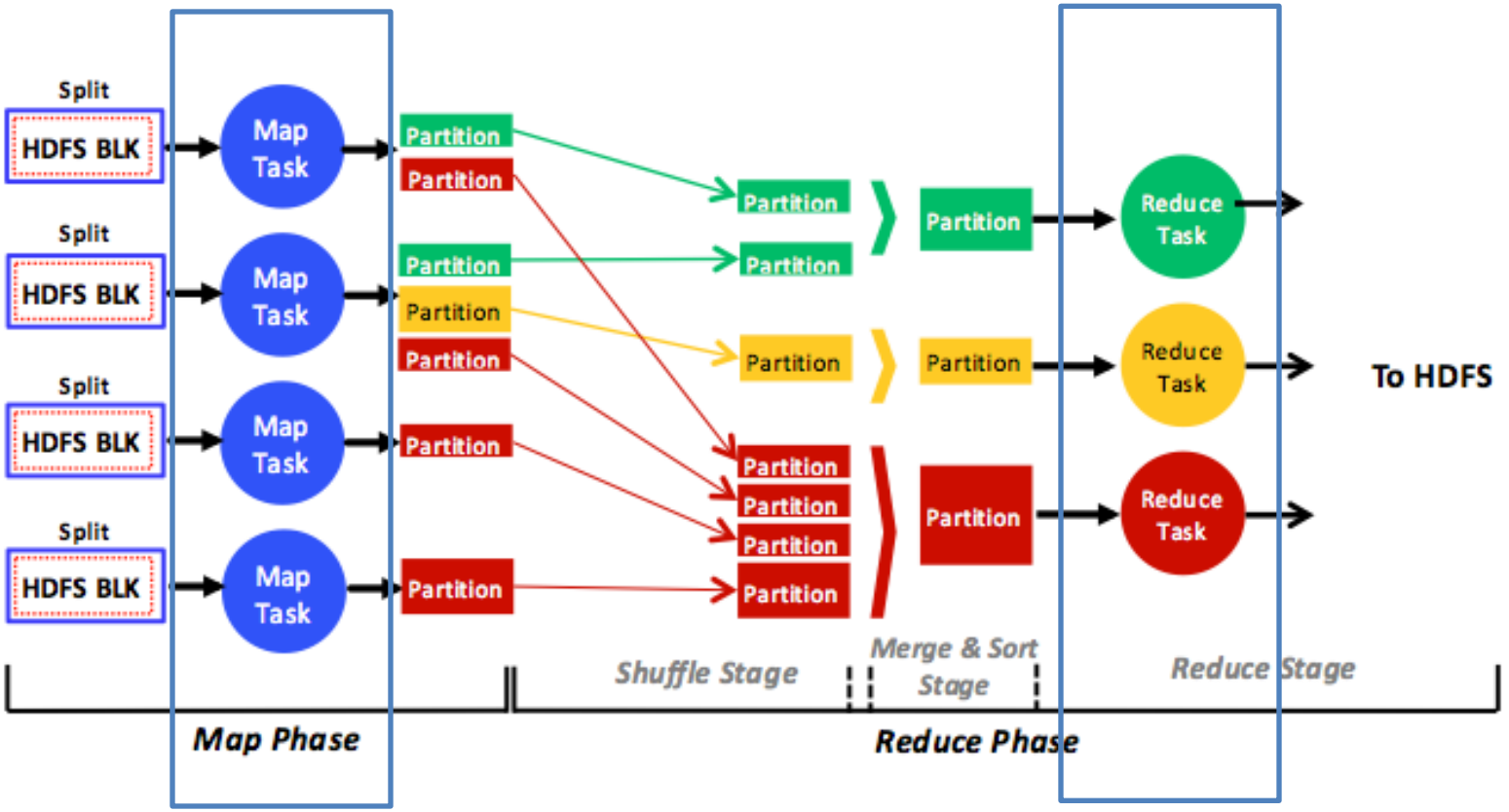
MapReduce Example



Steps of MapReduce



What you should write in EMR



Your own mapper

Your own reducer

Steps of MapReduce

- Map
- Shuffle
- Reduce
- Produce final output

Steps of MapReduce

- Map
  - Prepare input for mappers
    - Split input into parts and assign them to mappers
  - Map Tasks
    - Each mapper will work on its portion of the data
    - Output: **key-value pairs**
      - Keys are used in Shuffling and Merge to find the Reducer that handles it
      - Values are messages sent from mapper to reducer

Steps of MapReduce

- Shuffle
  - Sort the output of mapper by key
  - Split keys and assign them to reducers
  - Each key will be assigned to exactly one reducer
- Reduce
  - Each reducer will work on his part
  - Input: mapper's output (key-value pairs)
  - Output: the result needed
    - Different aggregation logic may apply

Steps of MapReduce

- Produce final output
  - Collect all output from reducers
  - Sort them by key

Parallelism in MapReduce

- Mappers run in parallel, creating different intermediate values from input data
- Reducers also run in parallel, each working on different keys
- However, reducers can not start until all mappers finish

Project 1, Module 2

- Processing sequentially can be limiting, we must:
  - aggregate the view counts and
  - generate a daily timeline of page views for each article we are interested in.
- Process a large dataset (~65 GB compressed)
- Setup an Elastic MapReduce job Flow
- Write simple Map and Reduce programs in the language of your choice
- You will understand some of the key aspects of Elastic MapReduce and run an Elastic MapReduce job flow.
- **Note:** For this checkpoint, assign the tag with
  - Key: Project and Value: 1.2 for all resources

Upcoming Deadlines

- Project Module: Elastic MapReduce
  - Due Sunday, February 2nd, 11:59PM Pittsburgh
- Checkpoint Quiz: Unit 2: Data Centers
  - Due Thursday, February 6th, 11:59PM Pittsburgh

Discussions

- Questions?

Demos

- Wordcount in MapReduce
- Introduction to EMR

Hadoop Streaming

Wordcount Demo

Outline

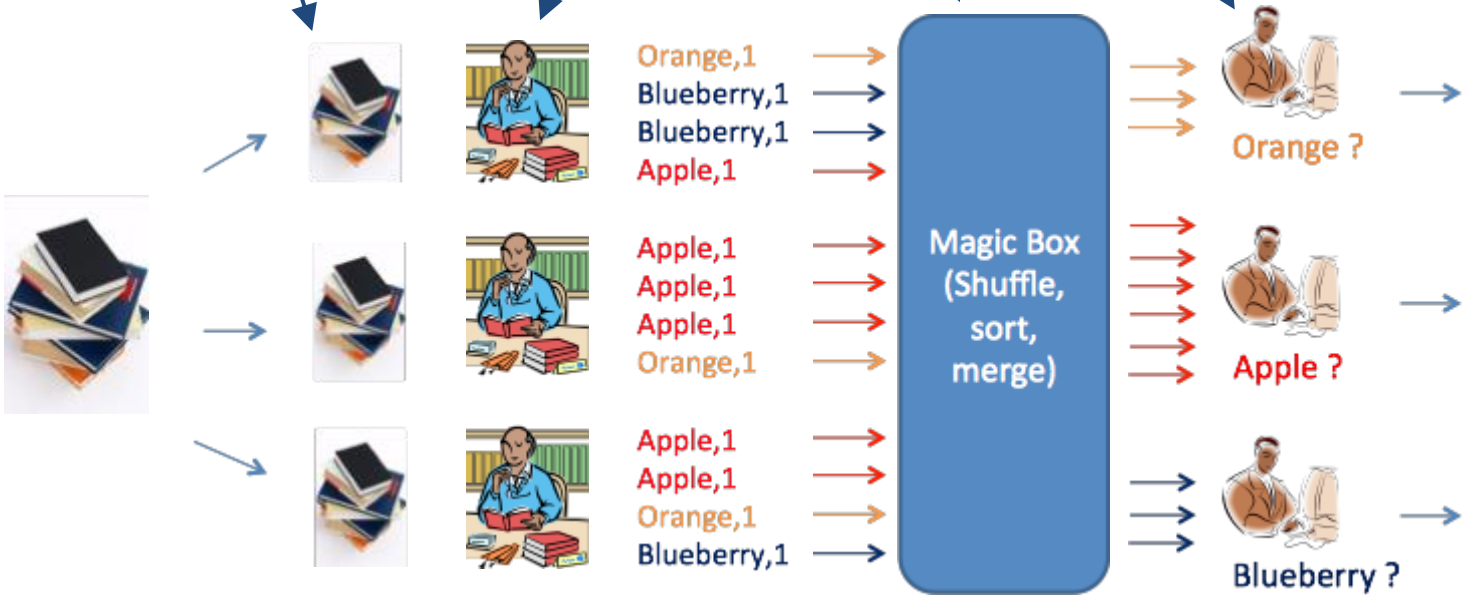
- Demo1:
  - Introduction to Hadoop streaming
  - Sample code for wordcount
- Demo2:
  - Run wordcount with Amazon EMR console

Hadoop Streaming

- Represent streaming via unix pipes (locally)

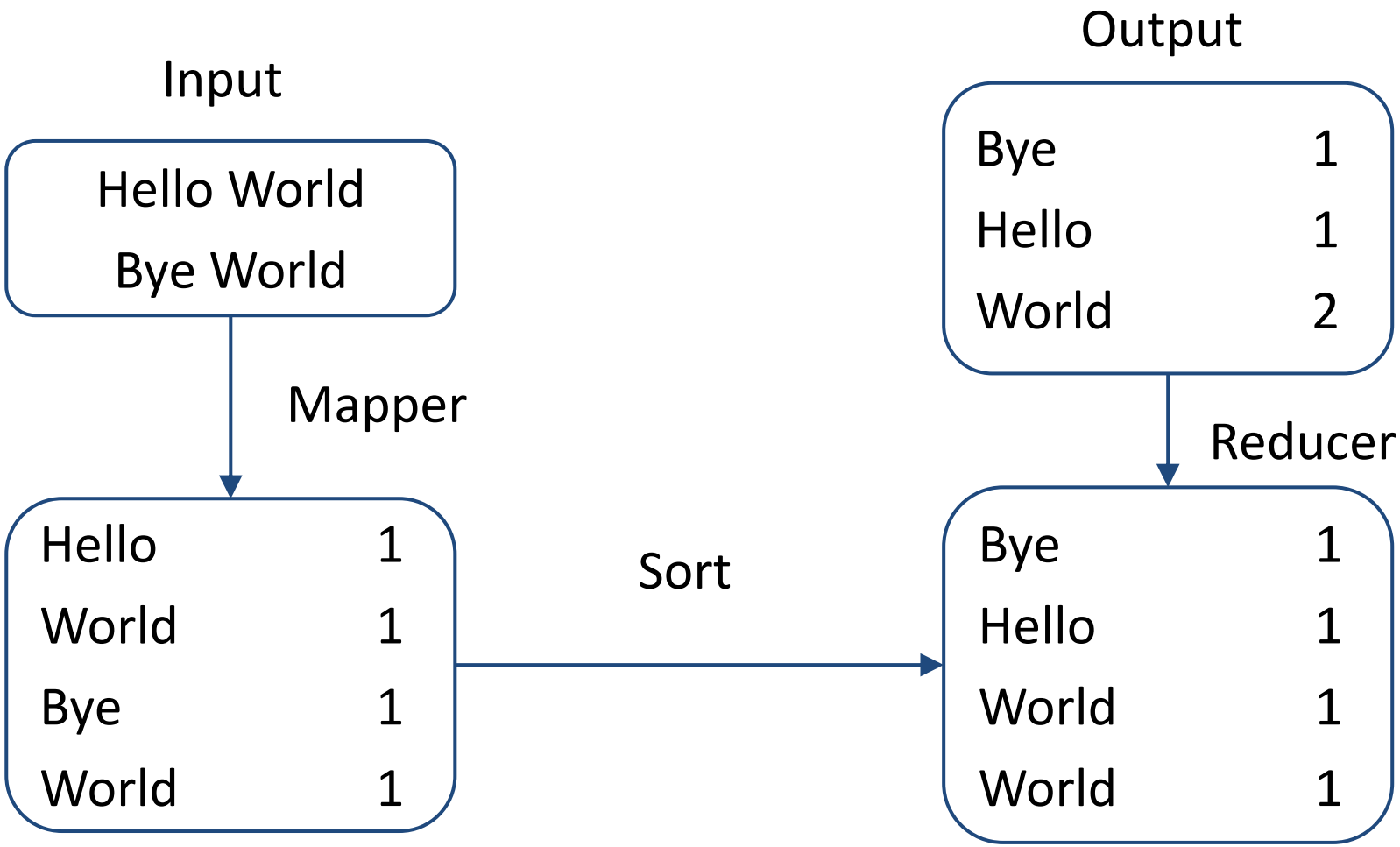
```
cat input.txt | mapper.py | sort | reducer.py
```

- How does it work in Mapreduce?



WordCount

WordCount


Input
Hello World
Bye World
Mapper
Hello 1
World 1
Bye 1
World 1
Sort
Output
Bye 1
Hello 1
World 2
Reducer
Bye 1
Hello 1
World 1
World 1

WordCount: wc-mapper.py

s3://wc-demo/wc-mapper.py

```
1. #!/usr/bin/python
2.
3. import sys
4. import re
5.
6. def main(argv):
7.         pattern = re.compile("[a-zA-Z][a-zA-Z0-9]*")
8.         for line in sys.stdin:
9.         for word in pattern.findall(line):
10.                 print word.lower() + "\t" + "1"
11.
12.if __name__ == "__main__":
13.         main(sys.arg
```

Remember to add this!!

1. Read line by line

2. Tokenize

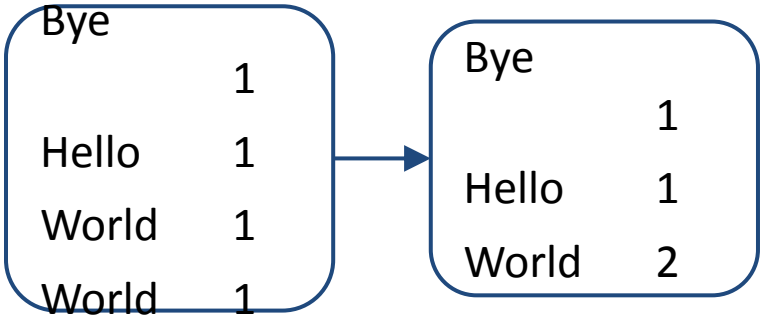
3. Generate (key, value) pair

Hello World
Bye World

Hello 1
World 1
Bye 1
World

WordCount: wc-reducer.py

s3://wc-demo/wc-reducer.py




```
6. (last_word, sum) = (None, 0)
7.
8. # input comes from STDIN
9. for line in sys.stdin:
10.     # parse the input we got from mapper.py
11.     (cur_word, value) = line.strip().split('\t')

6.     if last_word and last_word != cur_word:
7.         # write result to STDOUT
8.         print last_word + '\t' + str(sum)
9.         (last_word, sum) = (cur_word, int(value))
10.     else:
11.         (last_word, sum) = (cur_word, sum +
   int(value))

6. if last_word:
7.     print last_word + '\t' + str(sum)
```

Resources

- Python Tutorial http://docs.python.org/2/tutorial/
- Elastic MapReduce http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-what-is-emr.html
- Spot instance:
  - http://aws.amazon.com/ec2/spot-instances/