CS15-319 / 15-619
Cloud Computing

Recitation 3

September 9th & 11th, 2014

Overview

- Last Week's Reflection

  --Project 1.1, Quiz 1, Unit 1

- This Week's Schedule

  --Unit2 (module 3 & 4), Project 1.2

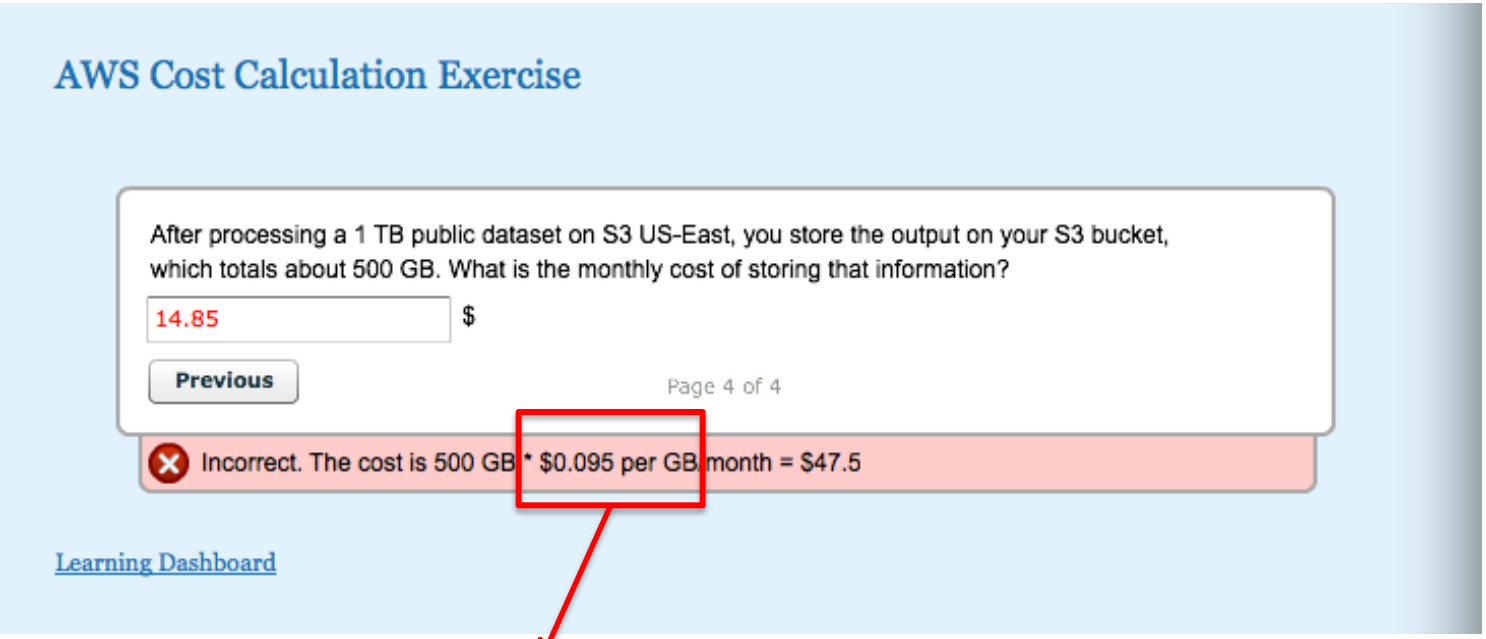
- Questions

Last Week Reflection

- Unit1 : Introduction to cloud computing
- Quiz 1
- Project 1.1
  - Explore the Wikimedia data set to learn the format
  - Sequentially parse and filter the data
  - Sort the data and save the output to a file
  - Extract & summarize some useful data by answering questions

Update in Unit 1

- "learn by doing" question 4 on page 137
  - S3 prices history : S3 Historical Price
  - AWS monthly calculator
  - Fixed on OLI for new viewers only



Stale price! Current price is 0.003 per GB/month

Quiz 1

- Review
  - Hybrid clouds and security
  - Utilization
  - Software service models

Project 1 Checkpoint 1

- Introduction to Big Data:
  - Sequential Analysis
    - Not running on AWS
      - Big penalty
    - Submission to s3 bucket
      - Follow guidelines on Piazza @218
    - Including credentials in code
      - Big penalty
    - Not tagging instances
    - Going over budget
      - Free tier not available for consolidated accounts

This Week's Schedule

- Complete Module 3 & 4 in Unit 2
  - Read all pages in modules:
    - Module 3: Data Center Trends
    - Module 4: Data Center Components
  - Complete activities on each page
    - In-module activities are not graded but for self-test
  - If you encounter a bug in the OLI write-up
    - provide feedback at the end of each OLI page
- Complete Project 1.2 (Elastic MapReduce)
  - Deadline, Sunday, September 14, 11:59pm EST

Why Study Data Centers ?

Data centers are your new computers!

- Make sure to read and understand the content of Unit 2
  - Equipment in a data center
  - Power, cooling, networking
  - How to design data centers
  - What could break
    - All software layers are on top of physical resources

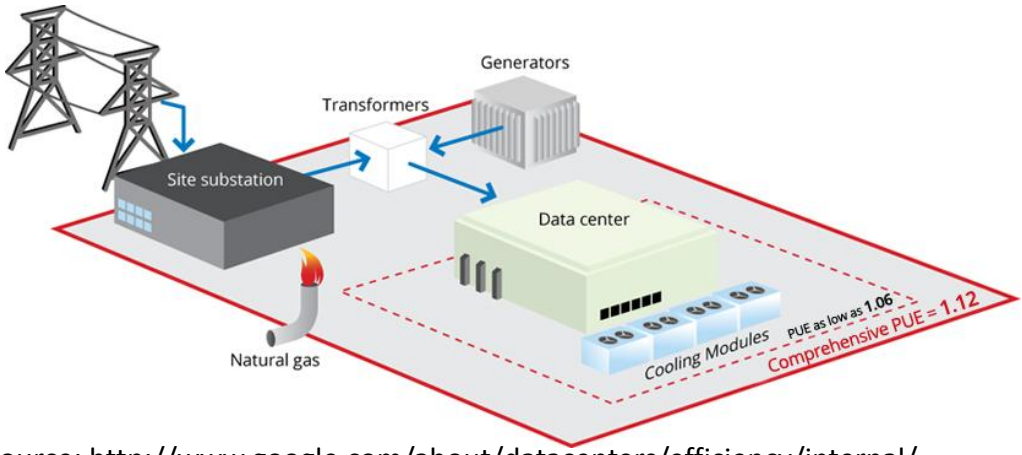
Module 3: Data Center Trends

- Definition & Origins
  - Infrastructure dedicated to housing computer and networking equipment, including power, cooling, and networking.
- Growth
  - Size (No. of racks and cabinets)
  - Density
- Efficiency
  - Servers
  - Server Components
  - Power
  - Cooling

Facebook data center

Module 4: Data Center Components

- IT Equipment
  - Anything that is mounted in a stack
  - Servers : rack-mounted
    - Motherboard
    - Expansion cards
  - Storage
    - Direct attached storage (DAS)
    - Storage area network (SAN)
    - Network attached storage (NAS)
  - Networking
    - Ethernet, protocols, etc.
- Facilities
  - Server room
  - Power (distribution)
  - Cooling
  - Safety



Source: http://www.google.com/about/datacenters/efficiency/internal/

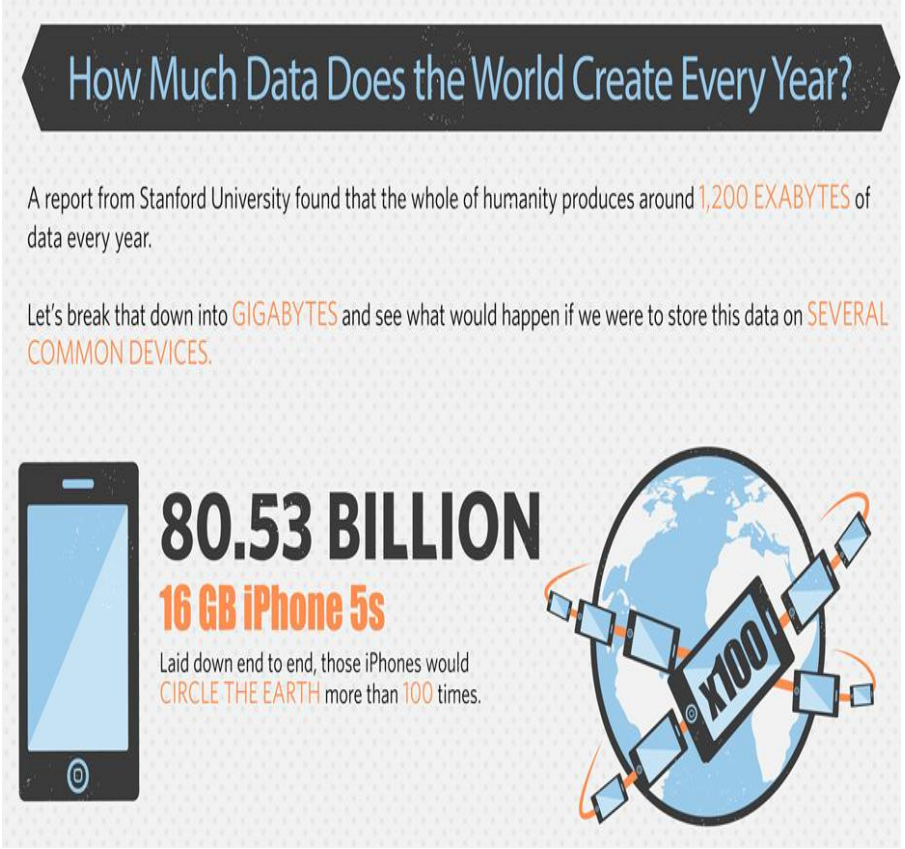
Motivation for MapReduce

- 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
  - Large–scale data processing
  - Out of memory
  - Slow
  - How would you deal with it?
    - Partition the input?
    - Distribute the work?
    - Coordinate the effort?
    - Aggregate the results?

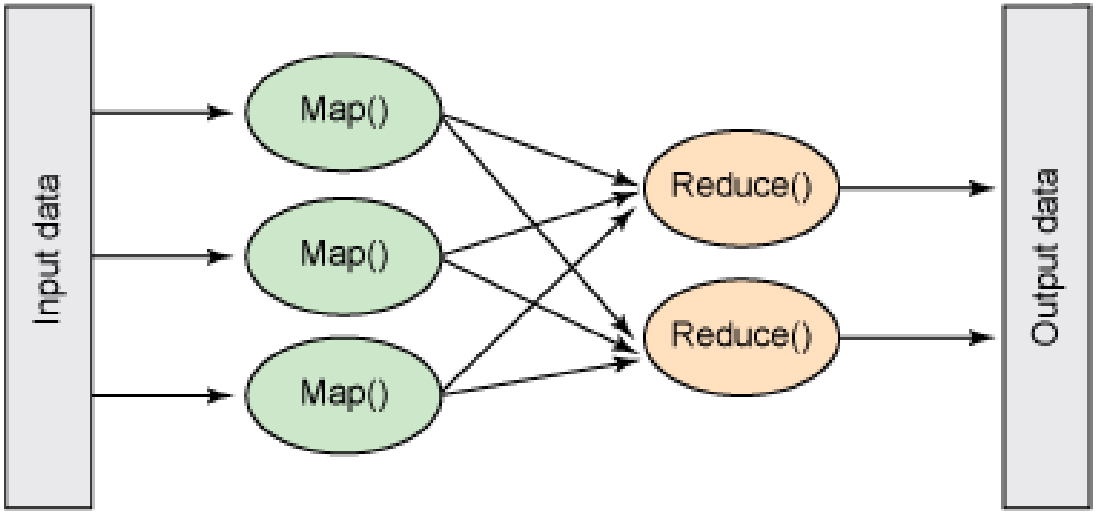


Motivation for MapReduce

- Google Example
- 20+ billion web pages x 20KB = **400+ TB**
- **~1,000** hard drives to store the web
- 1 computer reads 30-35 MB/sec from disk
  - **~4 months** to read the web
- Takes even more to **do** something useful with the data!
- So standard architecture for such problems emerged
  - Cluster of commodity Linux nodes
  - Commodity network (Ethernet) to connect them
- Google's computational / data manipulation model

Introduction of MapReduce

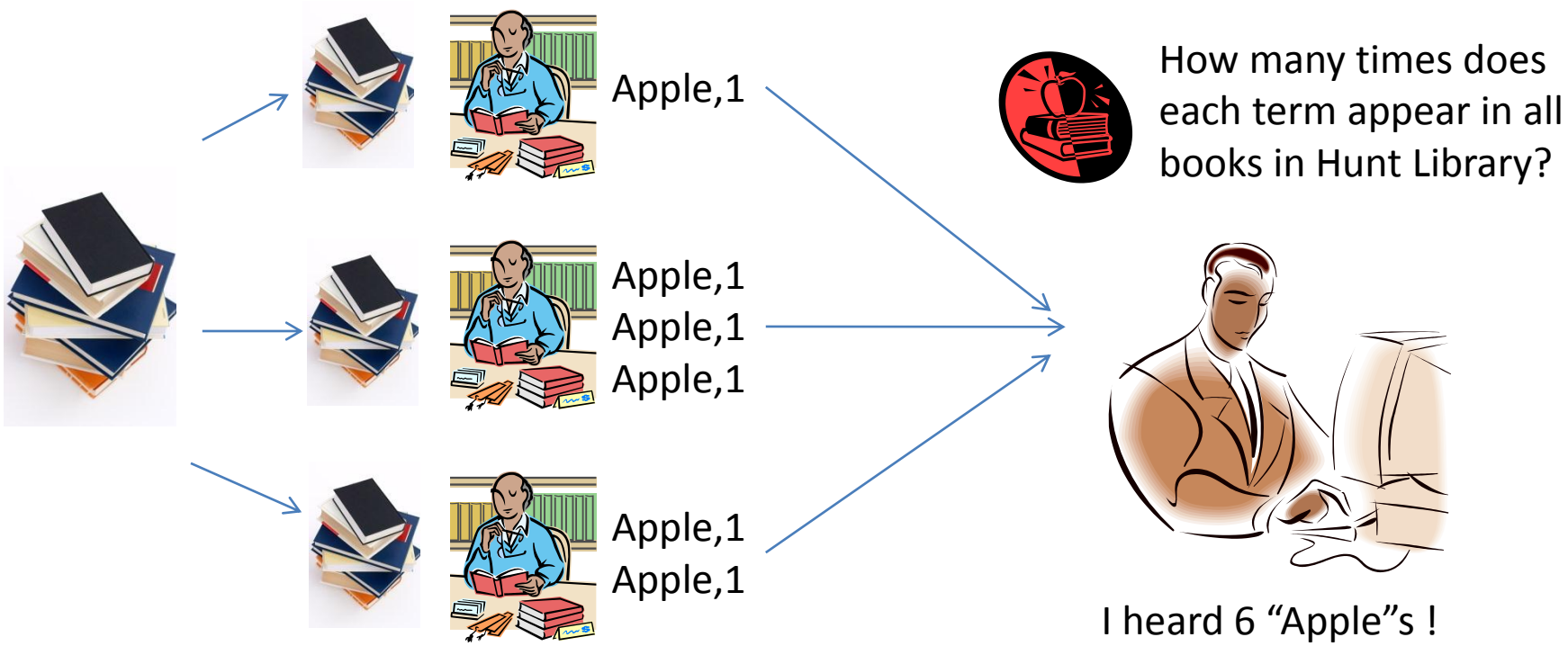
- **Definition**: Programming model for processing large data sets with a parallel, distributed algorithm on a cluster
- **Map**: Extract something you care about
- **Group by key**: Sort and Shuffle
- **Reduce**: Aggregate, summarize, filter or transform
- **Output** the result



MapReduce in 15-319/619

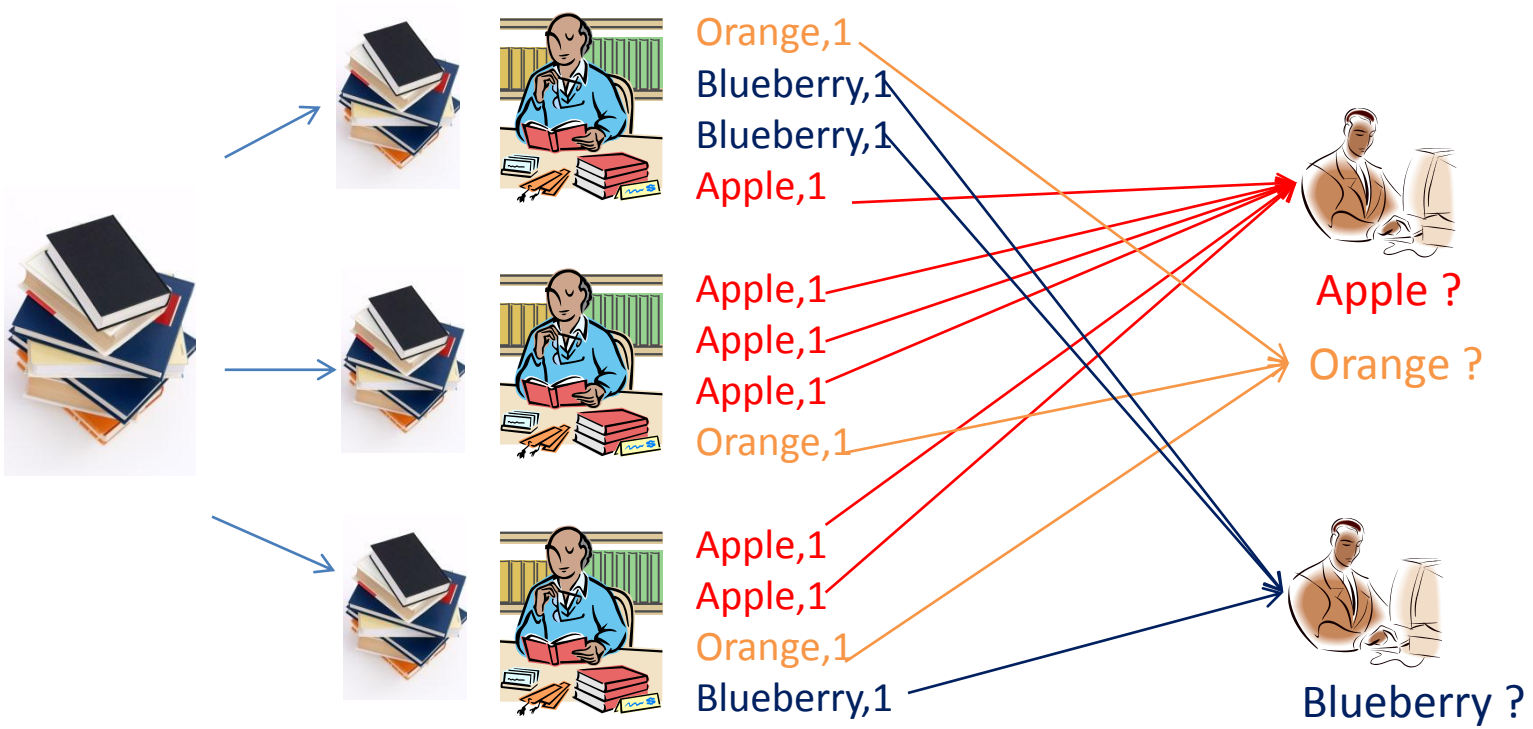
- In this course we are going to use MapReduce on 2 Platforms:
  1. Amazon Elastic MapReduce (this week)
  2. Hadoop (later projects)
     - Learn more about Hadoop MapReduce here

MapReduce Example



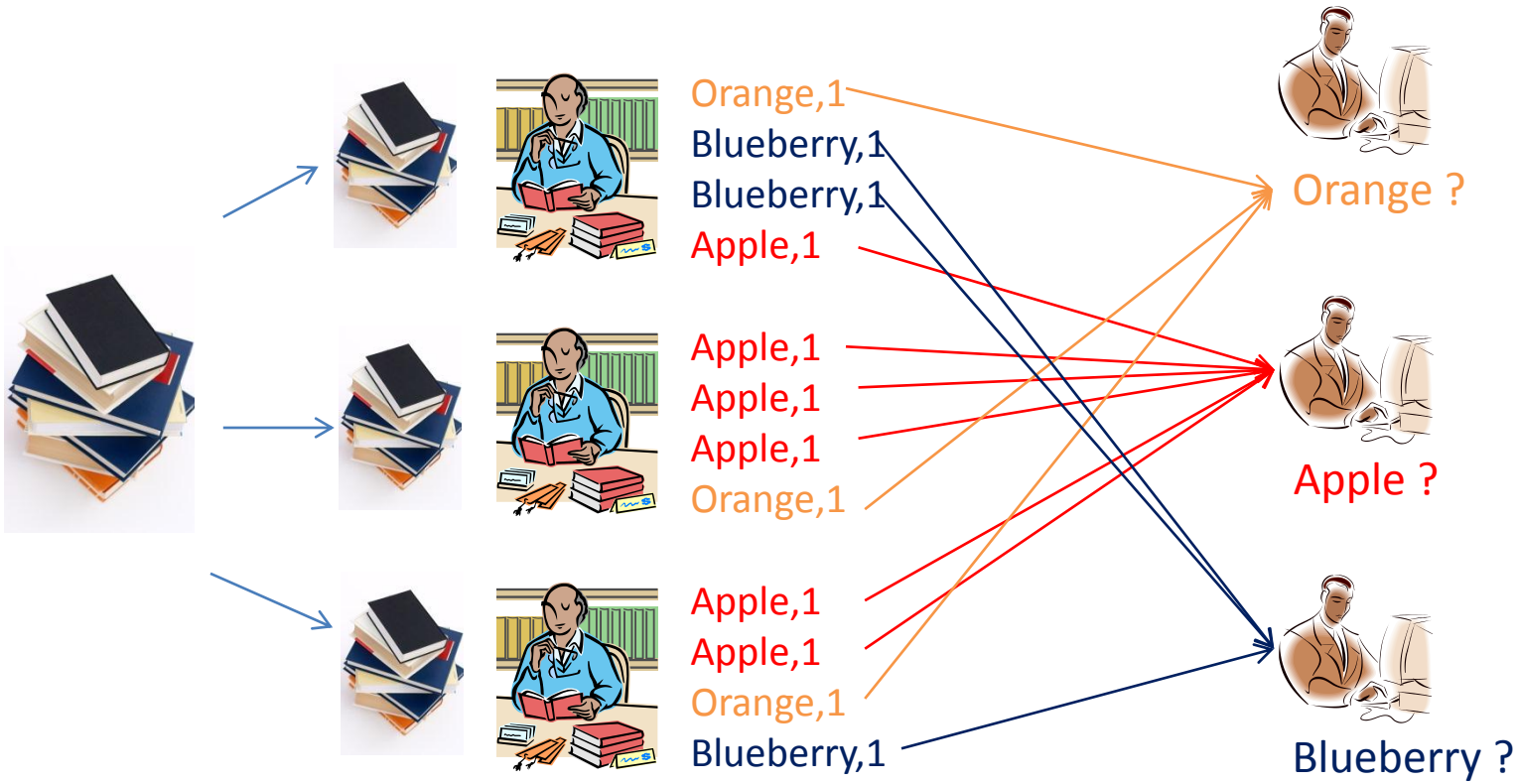
MapReduce Example

We can have two reducers
Each reducer can handle one or more keys

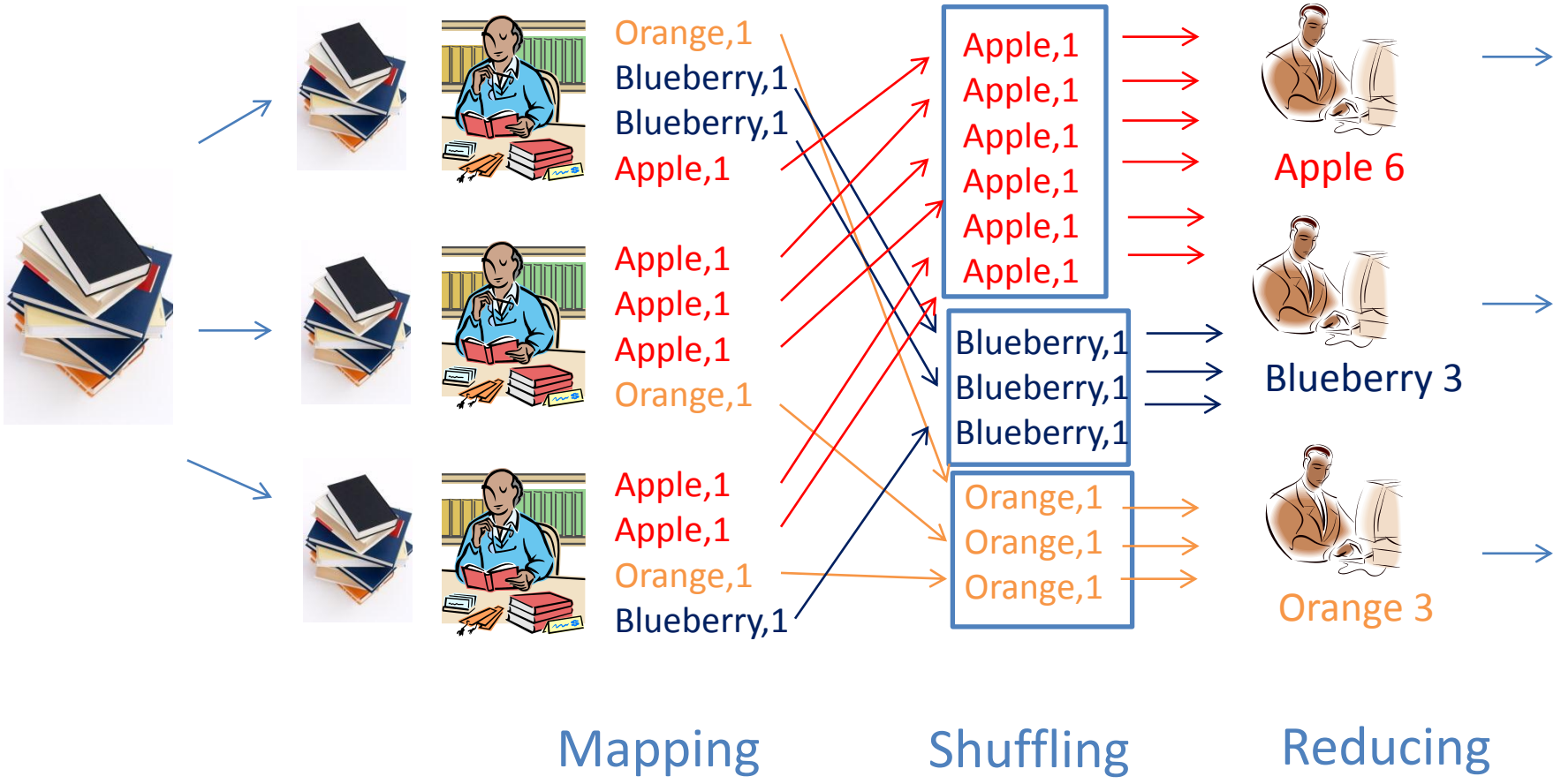


MapReduce Example

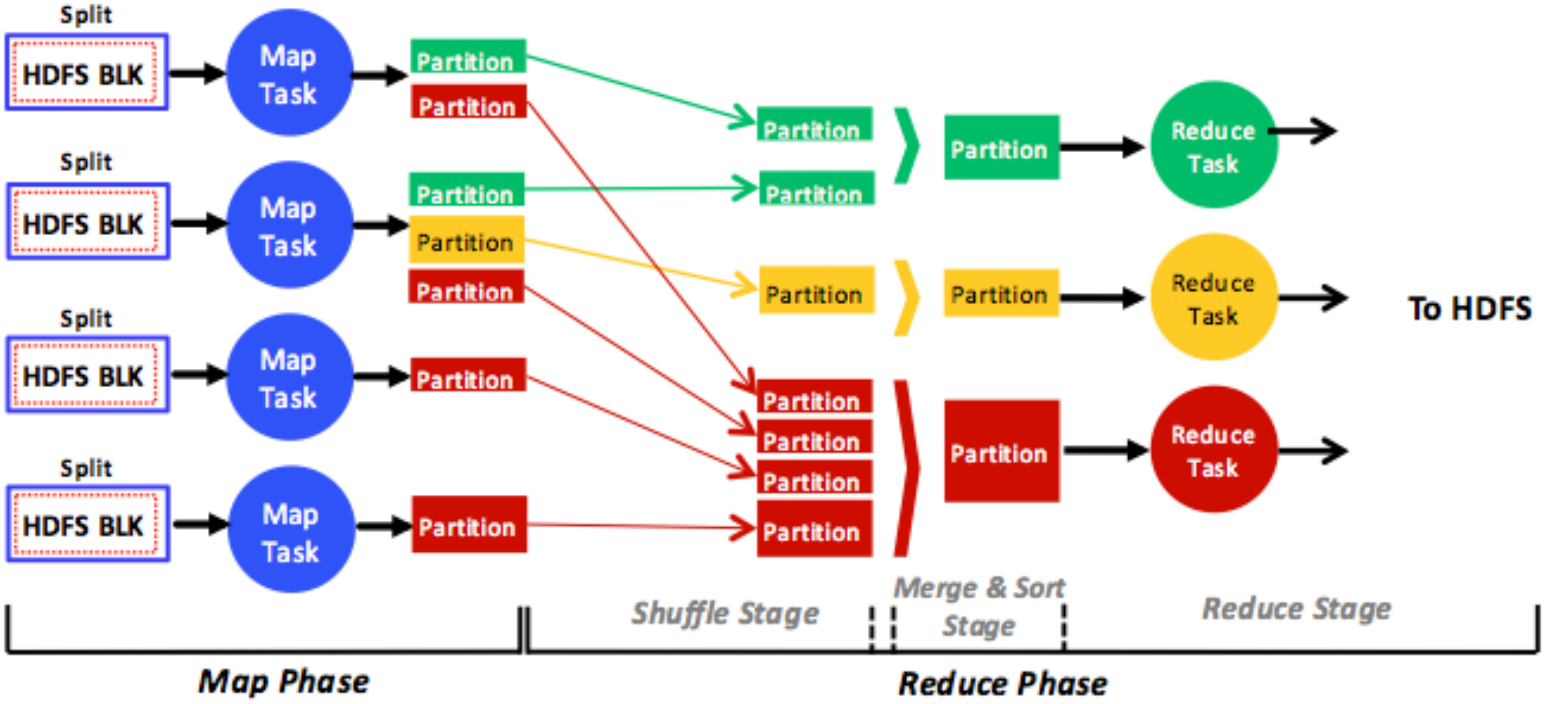
We can have three reducers



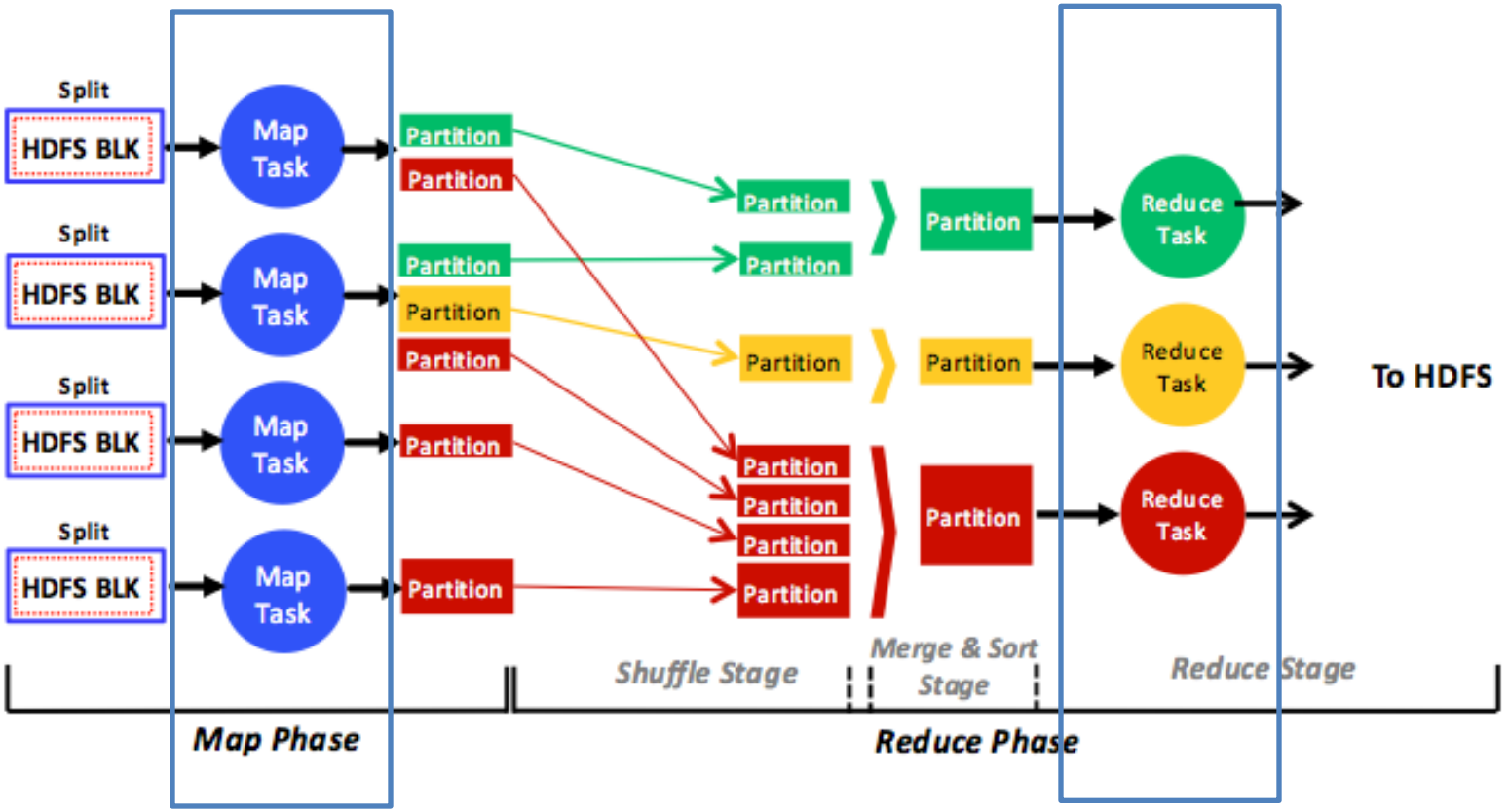
MapReduce Example



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
      - e.g. (Apple, 1)

Steps of MapReduce

- Shuffle
  - Group by key: sort the output of mapper by key
    - Split keys and assign them to reducers (based on hashing)
    - Each key will be assigned to exactly one reducer
- Reduce
  - Each reducer will work on one or more keys
  - Input: mapper's output (key-value pairs)
  - Output: the result needed
    - Different aggregation logic may apply
- Produce final output
  - Collect all output from reducers
  - Sort them by key

Mapreduce: Environment

- MapReduce environment takes care of:
  - **Partitioning** the input data
  - **Scheduling** the program's execution across a set of machines
  - Perform the **Group by key** (sort & shuffle) step
    - In practice, this is the bottleneck
  - Handling machine **failures**
  - Manage required inter-machine **communication**

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 cannot start until all mappers finish

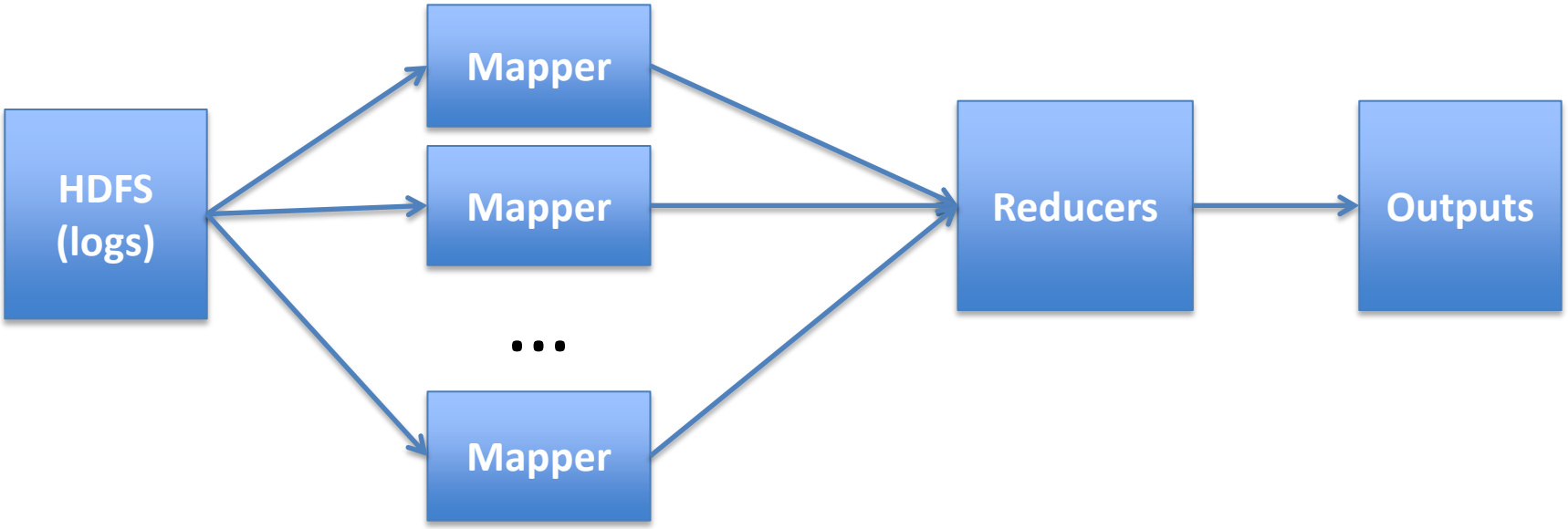
Real Example: Friend/Product Suggestion

- Facebook uses information on your profile, e.g. contact list, messages, direct comments made, page visits, common friends, workplace/residence nearness. This info is dumped into a log or a huge list, a **big data**.
- Then the logs are analyzed and put through a weighted matrix analysis and the connections which are above a threshold value are chosen to be shown to the user.

Real Example: Friend/Product Suggestion

Generate key-value pairs:
Key: Friends pair
Value: Friends statistics (e.g. common friends)
e.g.
(Tom, Sara statistics)

Aggregate the statistics value for the same key and output the friends pair if it's above the threshold
e.g.
(Tom Sara)



Project 1.2 Elastic MapReduce

- 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 (~70 GB compressed)
- Setup an Elastic MapReduce job Flow
- Write simple Mapper and Reducer 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

AWS Expenditure

- Monitor AWS expenses regularly
  - For this week:
    - EMR cost is **"on top of"** the EC2 cost of instance and EMR cost is fixed per instance type per hour
    - for example, m2.4xlarge EMR cost is $0.42 on-top-of the spot pricing ($0.14)
    - Check out AWS EMR pricing !
  - Suggestions
    - Terminate your instance when not in use
      - stop still costs EBS money!
    - Use smaller instances to test your code
    - Use **spot instances** to save cost
    - Use small sample dataset in EMR to decide the **instance type** **and number of instances** to use

IMPORTANT!

AWS Expenditure

- Tagging AWS resources
- Setting up CloudWatch billing alert
- Using spot instances

Please refer to slides in Recitation 2!

Penalties

- If
  - No tag ➔ **10%** penalty
  - Expenditure > project budget ➔ **10%** penalty
  - Expenditure > project budget * 2 ➔ **100%** penalty
  - Copy any code segment from ➔ lowest penalty is **200%**
    - Other students
    - Previous students
    - Internet (e.g. Stackoverflow)
- Do not work on code together
  - This is about learning
  - If you do, we *will* find out and take action

Submitting Code

- Submit your code on S3, by the deadline
  - Submitting the wrong S3 URL on OLI will be penalized
  - Submitting an incorrectly configured bucket will be penalized (see the instructions in the Project Guidelines page in the Project Primer on OLI)
- We will manually grade all code
  - Be sure to make your code readable
    - Preface each function with a header that describes what it does
    - Use whitespace well.
      Indent when using loops or conditional statements
    - Keep each line length to under 80 characters
    - Use descriptive variable names
    - For more detail, please refer to www.cs.cmu.edu/~213/codeStyle.html
  - If your code is not well documented and is not readable, we will deduct points
    - Documentation shows us that you know what your code does!
    - The idea is also NOT to comment every line of code

S3 Code Submission Guidelines

- Make your submission a single zip file (.zip) with name "**project<no>_AndrewID_q<no>"**.
- Pack all your code, in ".java", ".py", ".rb", ".sh" or other formats in the zip file.
- Do **NOT** submit associated libraries and binary files (.jar and .class files).
- Please do **NOT** submit multiple s3 URLs in the text field on OLI.
- Create a single submission bucket for all of your code, using the folder hierarchy illustrated in the project primer on page 151 on OLI.
- Do NOT submit AWS Credential Files (aws.credentials) or any other files that contain your AWS Keys within your bucket.

  ➔ **10%** penalty
- Do not make your buckets public.

Questions?

Upcoming Deadlines

- Project 1: Introduction to Big Data Analysis
  - Elastic MapReduce
  - Checkpoint Available Now
  - Due 9/14/2014 11:59 PM EST
- Quiz 2: Data Centers
  - Not Available Yet
  - Due 9/18/2014 11:59 PM EST