CS15-319 / 15-619
Cloud Computing

Recitation 12

November 12th and Nov 15th, 2013

Announcements

- Encounter a general bug:
  - Post on Piazza
- Encounter a grading bug:
  - Post Privately on Piazza
- Don’t ask if my answer is correct
- Don’t post code on Piazza
- Search before posting
- Post feedback on OLI

Piazza Questions

- STDOUT, STDERR redirection
  - ./run.sh 1> result.out 2> error.out
- Round Trip Time (RTT)
  - Time between generating a request and receiving a response
- Timeout to HBase cluster
  - Security group
  - Time out between master node and slave nodes
    - Common under a heavy load
- DynamoDB throughput exceeded
  - Warning: should be normal
  - It might lead to earlier termination for YCSB (still investigating)

DynamoDB vs. HBase

- Data Model
  - Key-value vs. Column oriented Key-value
- Proprietary vs. Open source
- Cost
  - DynomoDB: Provisioned Throughput Capacity
  - HBase: Instance + EMR
- Limitations:
  - DynamoDB:
    - Item size: 64 KB
    - Query result: 1 MB

Project 3, Module 5 Reflections

- When to use DynamoDB:
  - Required throughput is determined
    - e.g. steady arrival rate
  - Easier to implement and scale
  - Enough budget
    - Charged by provisioned throughput capacity
- When to use HBase:
  - Low cost
  - Less constrains (Item size, query result)
  - Open source, more configurable

Module to Read

- UNIT 5: Distributed Programming and Analytics Engines for the Cloud
  - Module 16: Introduction to Distributed Programming for the Cloud
  - Module 17: Distributed Analytics Engines for the Cloud: MapReduce
  - Module 18: Distributed Analytics Engines for the Cloud: Pregel
  - Module 19: Distributed Analytics Engines for the Cloud: GraphLab

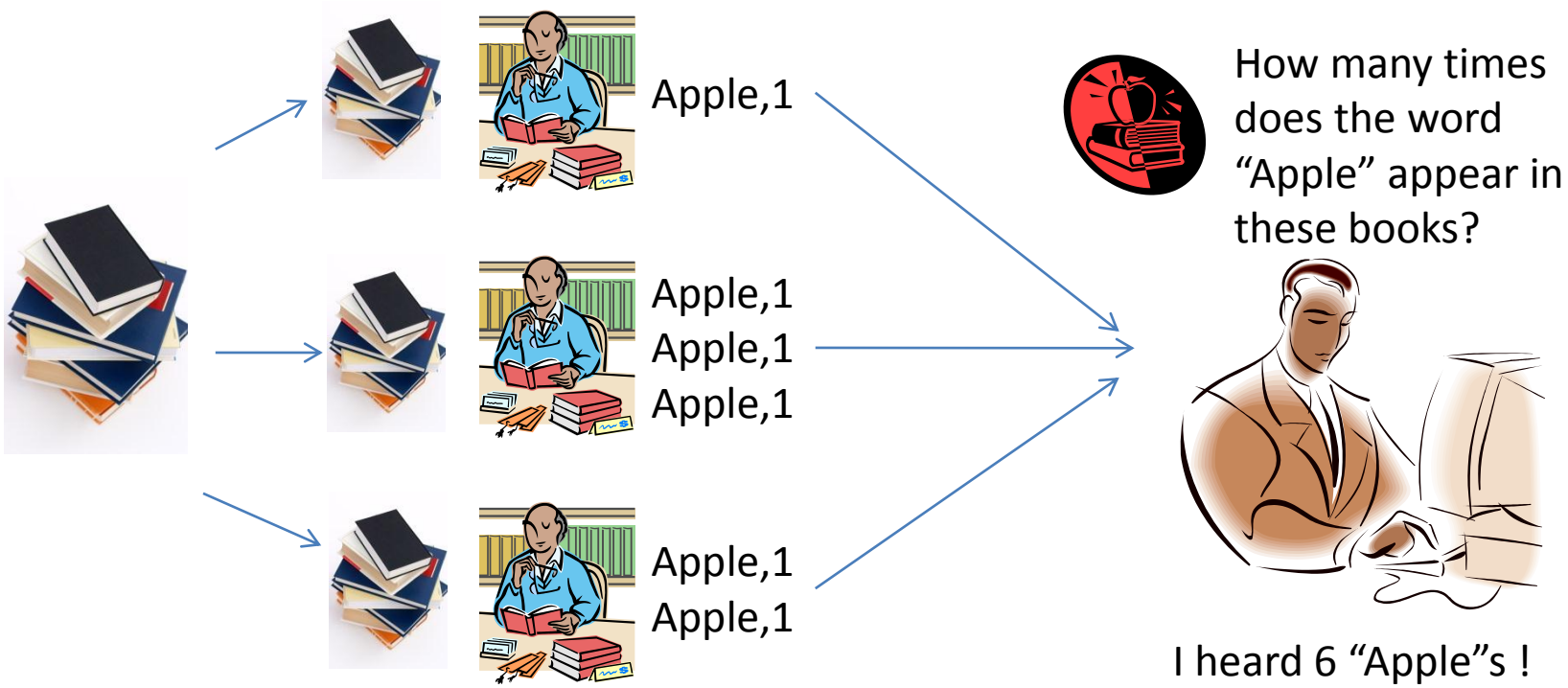
MapReduce

- The idea of MapReduce

Please tell me how many times does the word “Apple” appear in these books?

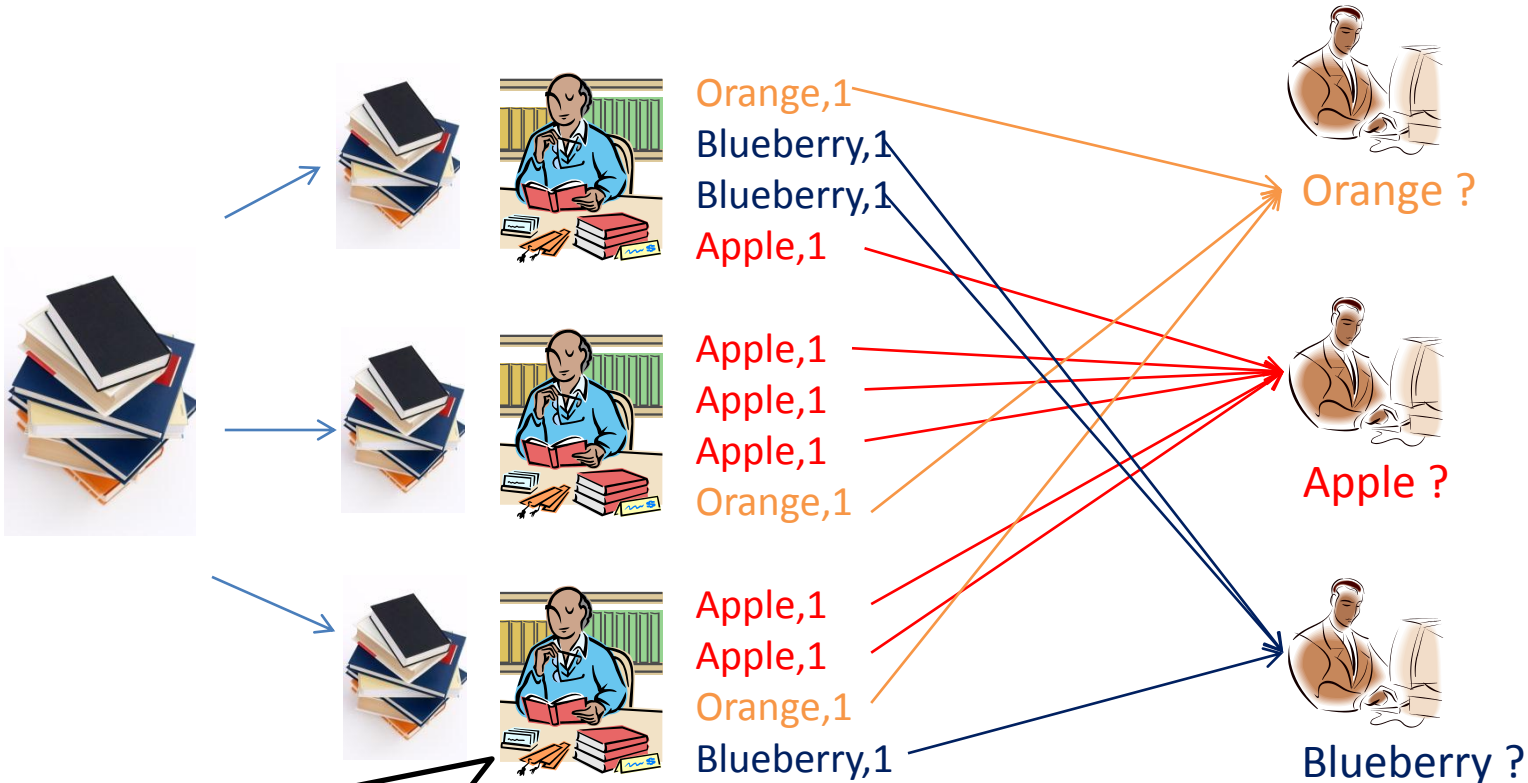
MapReduce

- The idea of MapReduce



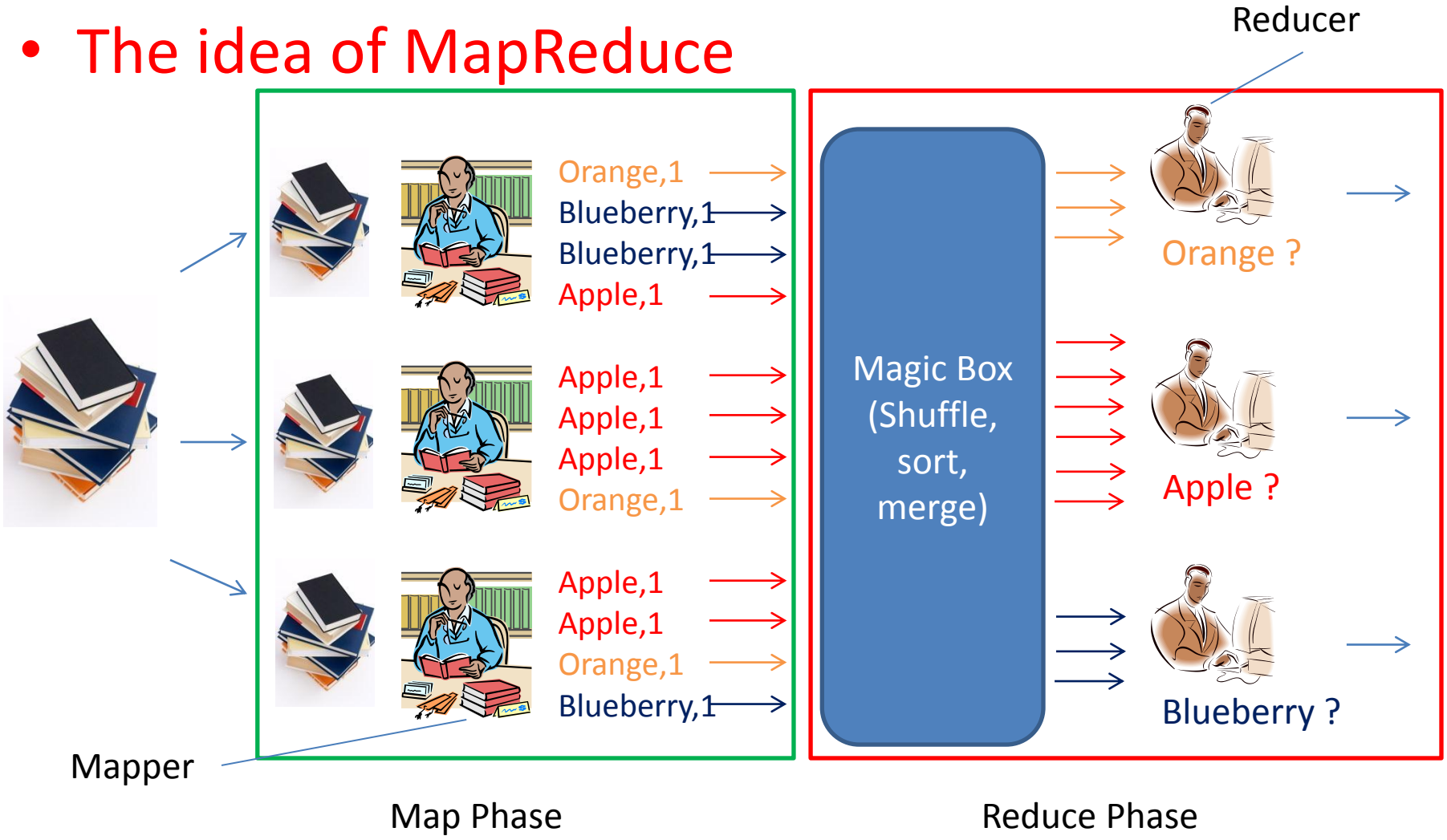
MapReduce

- The idea of MapReduce



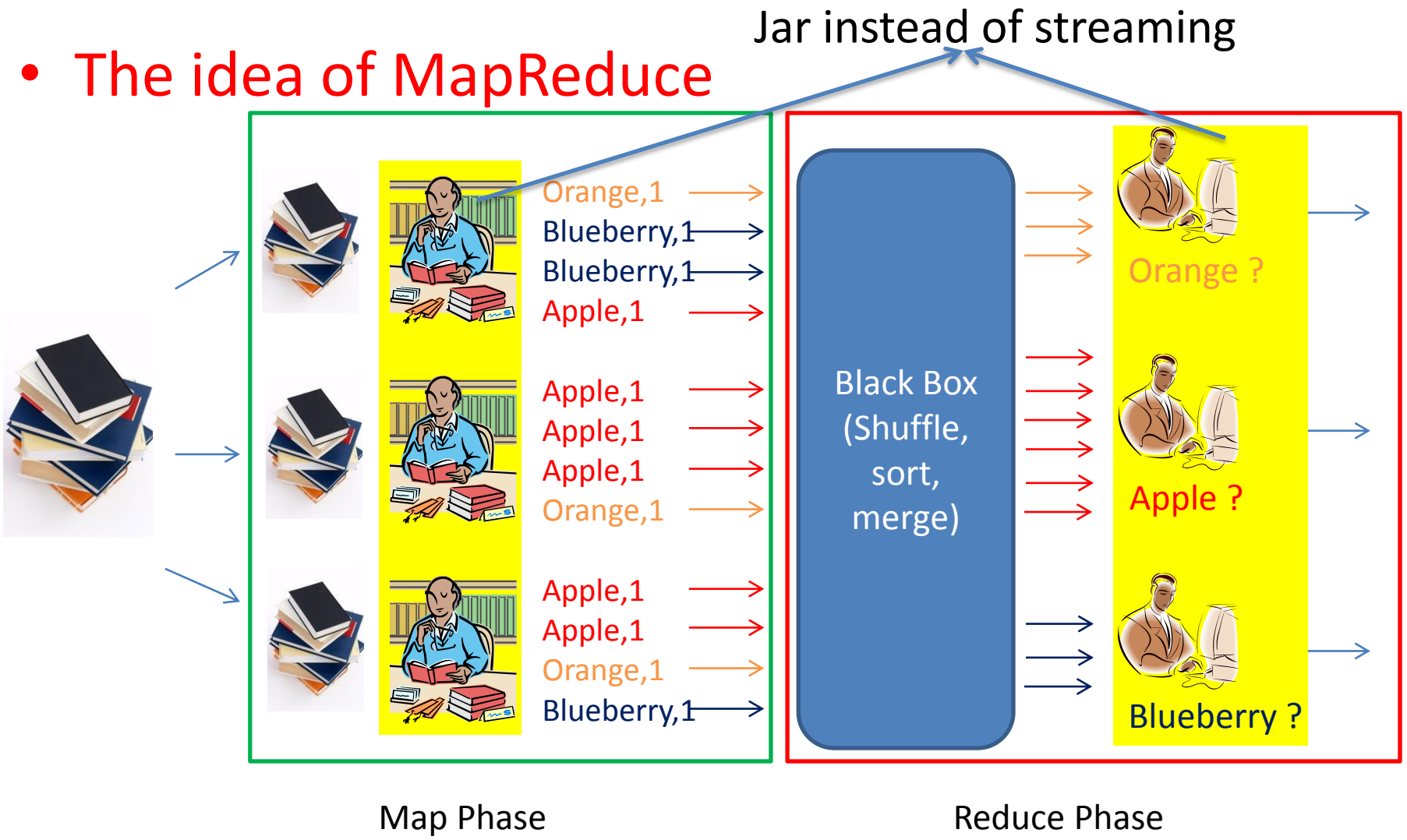
MapReduce

- The idea of MapReduce



MapReduce

- The idea of MapReduce



MapReduce

- Mapper
  - Input: lines in files in our project
  - Output: **key-value pairs**
    - **Keys** are used in Shuffling and Merge to find the Reducer that handles the intermediate output for that specific key. (in our example, Apple, Orange and Blueberry are keys)
    - **Values** are messages sent from mapper to reducer (in our case it is always 1)
    - Mappers' output is intermediate because reducers will receive the key-value pairs and take them as input.

MapReduce

- Reducer
  - Input: **key-value pairs**
  - Output: the final result we need
    - Depends on what we want, our code should process the value in the key-value pairs that we got accordingly (in the word count example, we just add up all the values).

Project 4 Module 1

- Write a MapReduce program that will build an inverted index of documents
- EMR Java (instead of streaming)

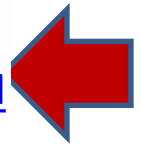
Recommendations

- Make sure to test for correctness with small datasets first
- EMR will charge you one hour of usage for instances even though your EMR job failed to start

Upcoming Deadlines

- Project 4:

Project 4		
MapReduce		
Hadoop MapReduce	Checkpoint	Available Now Due 11/17/13 11:59 PM



- Unit 5:

UNIT 5: Distributed Programming and Analytics Engines for the Cloud
Module 16: Introduction to Distributed Programming for the Cloud
Module 17: Distributed Analytics Engines for the Cloud: MapReduce

Demo Outline

- Code for MapReduce example
- Use EMR with customized jar