CS15-319 / 15-619 Cloud Computing

Recitation 3

Sept 15 & 17, 2015

Guest Lecture

Mark Russinovich
CTO Microsoft Azure
Thursday, 9/24
4:30pm in GHC 4307

Add it to your calendar!

Overview

- **Administrative Issues**
- **Last Week's Reflection**
  - Project 1.1, OLI Unit 1, Quiz1
- **This Week's Schedule**
  - Project1.2, OLI Unit 2, Module 3 and 4, Quiz2
- **Demo**
- **Questions**

Administrative

- TA office hours are posted
  - Piazza
  - Google calendar

- Suggestions for using Piazza
  - Discussion forum, contribute questions and answers
  - Read the Piazza Post Guidelines (@7) before asking
  - Read Piazza questions & answers carefully to avoid duplicate ones
  - Don’t ask a public question about a quiz question
  - Try to ask a public question if possible

Keeping Your Account Secure

- Do not make your code available publically on the internet
- Remove any account identification information away before committing to a private repository
- Do NOT submit .pem files through the autograder.
- Remove account credentials before submitting code

Last Week Reflection

- Reading:
  - Unit 1: Introduction to Cloud Computing
    - Module 2 : Cloud Building Blocks
  - Quiz 1: Introduction to Cloud Computing
- Project:
  - Project 1.1:
    - Wikipedia Dataset
    - Filtering one hour’s worth of data

FAQ P1.1

- Q: How to deal with the situation where the page does not have a title?
- A: Data from the real world is always dirty, and you should think about a proper way to handle data according to your needs (This will also be the case for the 15619Project). For Project 1.1, you may choose to filter this malformed record out.

- Q: Why do I get a grade of 0 for Q10 on P1.1?
- A: We will grade your code manually after the submission deadline and you can see our feedback then. You can use the feedback we give in order to write better code for the next project. The team needs one week to manually grade your code submissions and provide feedback.

Looking back at Project 1.1

- Loading all the data to memory to filter and process is a bad idea!
  - Recurring theme in the course projects
  - You will see this in 15619 Team Project (ETL)
- A better approach: work from disk, build a processing pipeline
  - Write programs that process the data line by line

This Week's Schedule

- Complete Unit 2 (Modules 3 & 4)
- **Quiz 2**
  - Deadline, Friday, Sept 18, 11:59pm ET
- **Complete Project 1.2 (Using Elastic MapReduce)**
  - Deadline, Sunday, Sept 20, 11:59pm ET

Why Study Data Centers in Unit 2?

- The cloud is the data centers
- Learn what influences
  - performance, failure, cost, …
- Make you a better cloud programmer
- 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

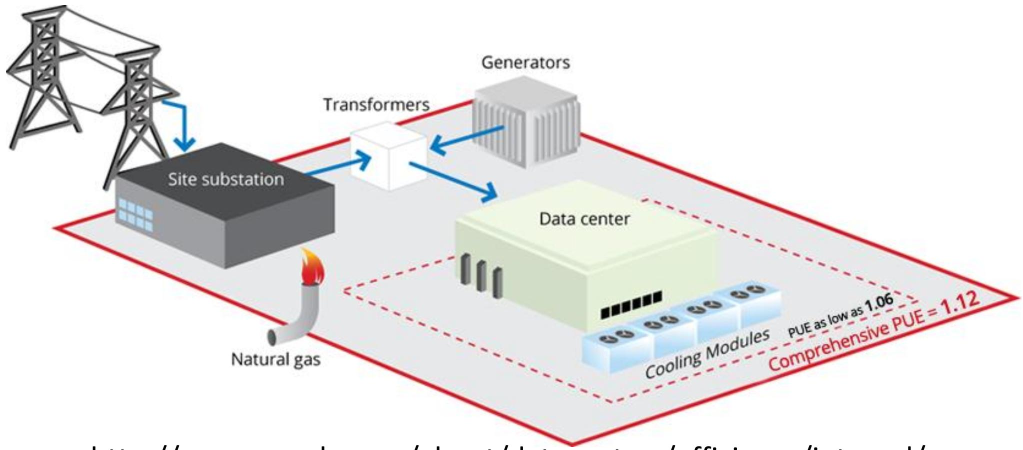
Module 3: Data Center Trends

- 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
  - Type of Storage
    - Direct attached storage (DAS)
    - Storage area network (SAN)
    - Network attached storage (NAS)
  - Networking
    - Ethernet, protocols, etc.
- Facilities
  - Server room
  - Power (distribution)
  - Cooling
  - Safety



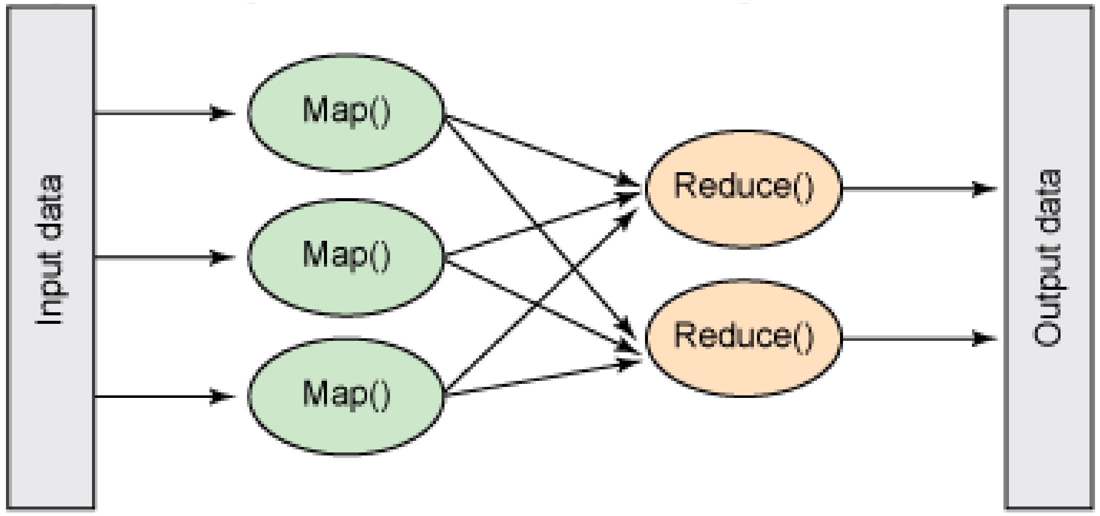
Source: http://www.google.com/about/datacenters/efficiency/internal/

Project 1.2

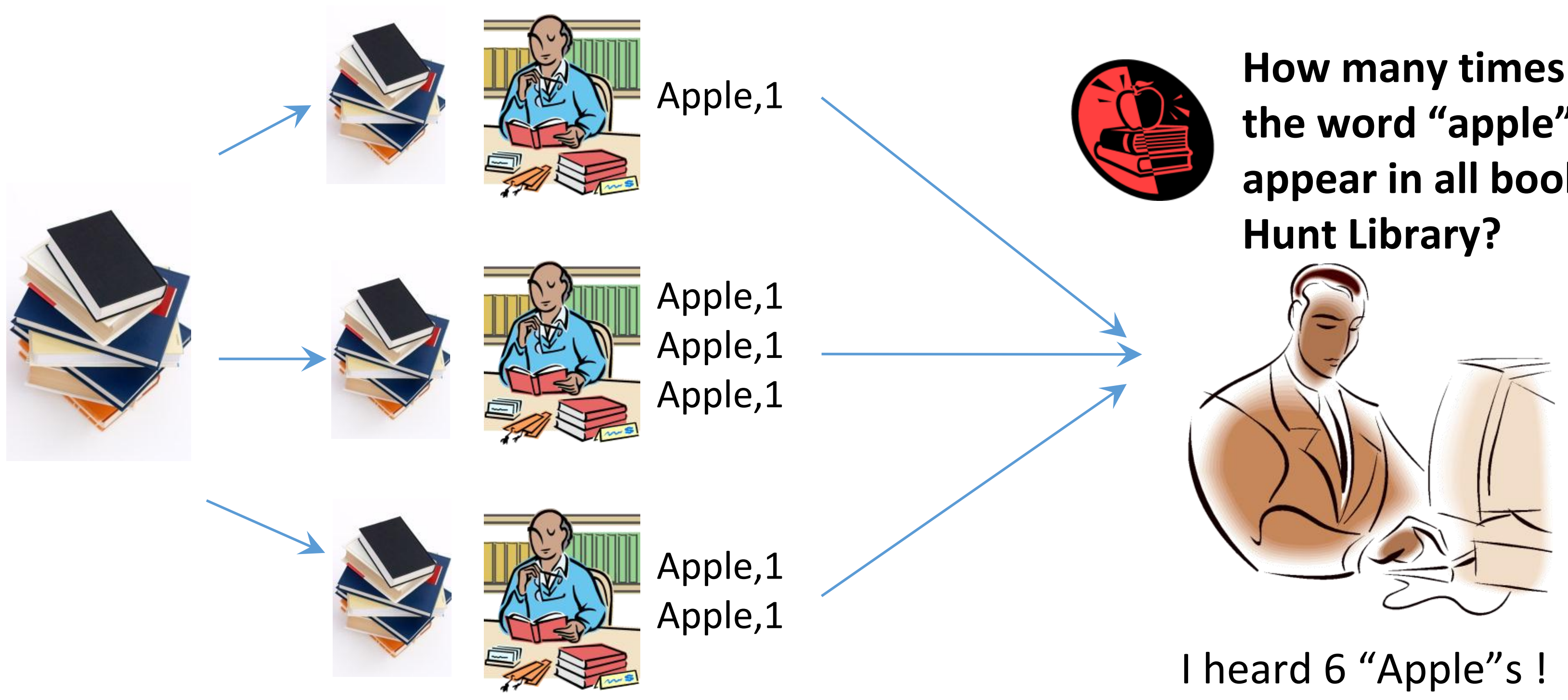
- In Project 1.1, we processed 1 hour of data on one single machine
- How do you filter and sort the data for one month?
    - 720 files total (**~ 70 GB compressed, ~300 GB uncompressed**)
- Parallel & Distributed Processing
    - How about Pthreads/MPI/…?
        - How simple are these frameworks?
        - Need to design many elements from scratch:
            - File Handling
            - Task Management
            - Orchestration
        - Painful. Take 15440/15618 for a taste ☺

Introduction to 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 Example

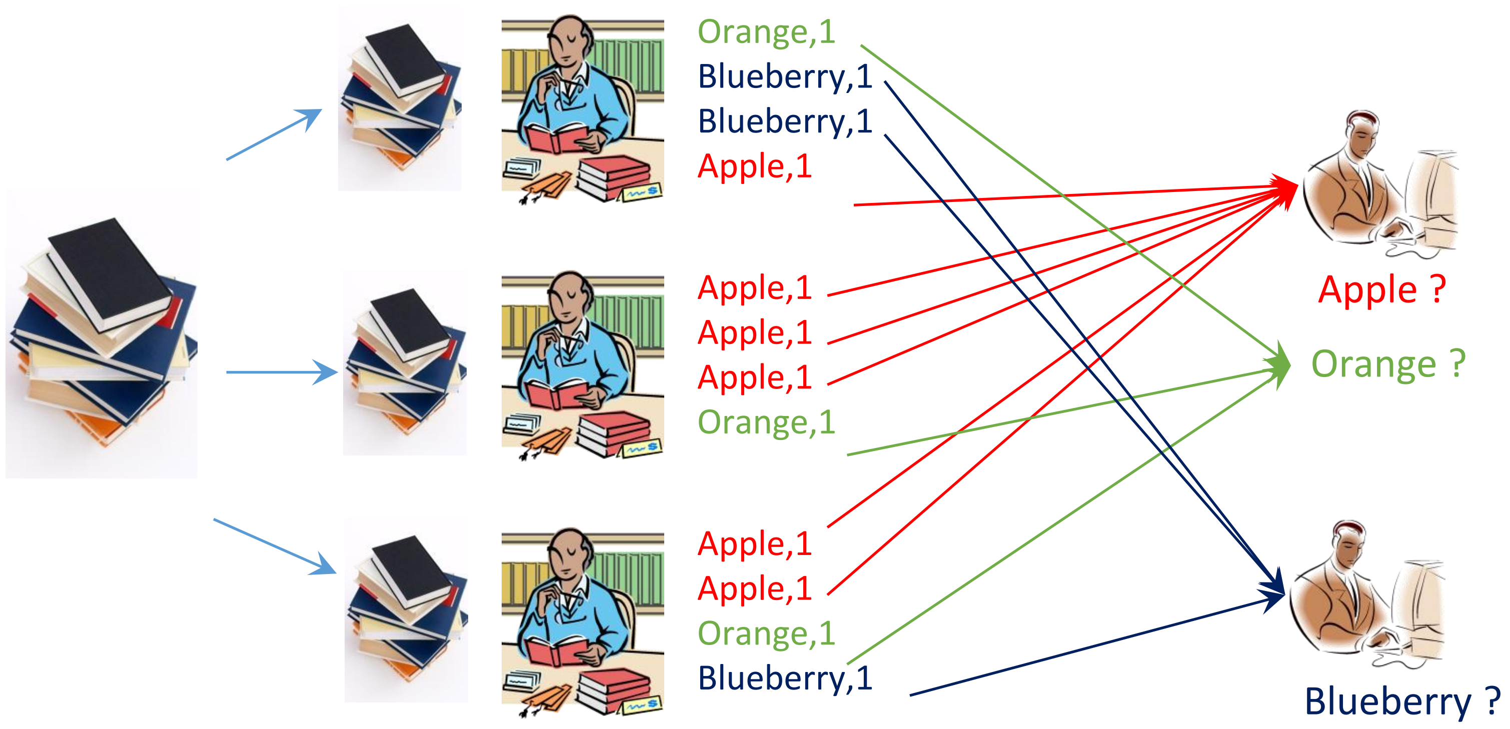


How many times does the word “apple” appear in all books in Hunt Library?

I heard 6 “Apple”s !

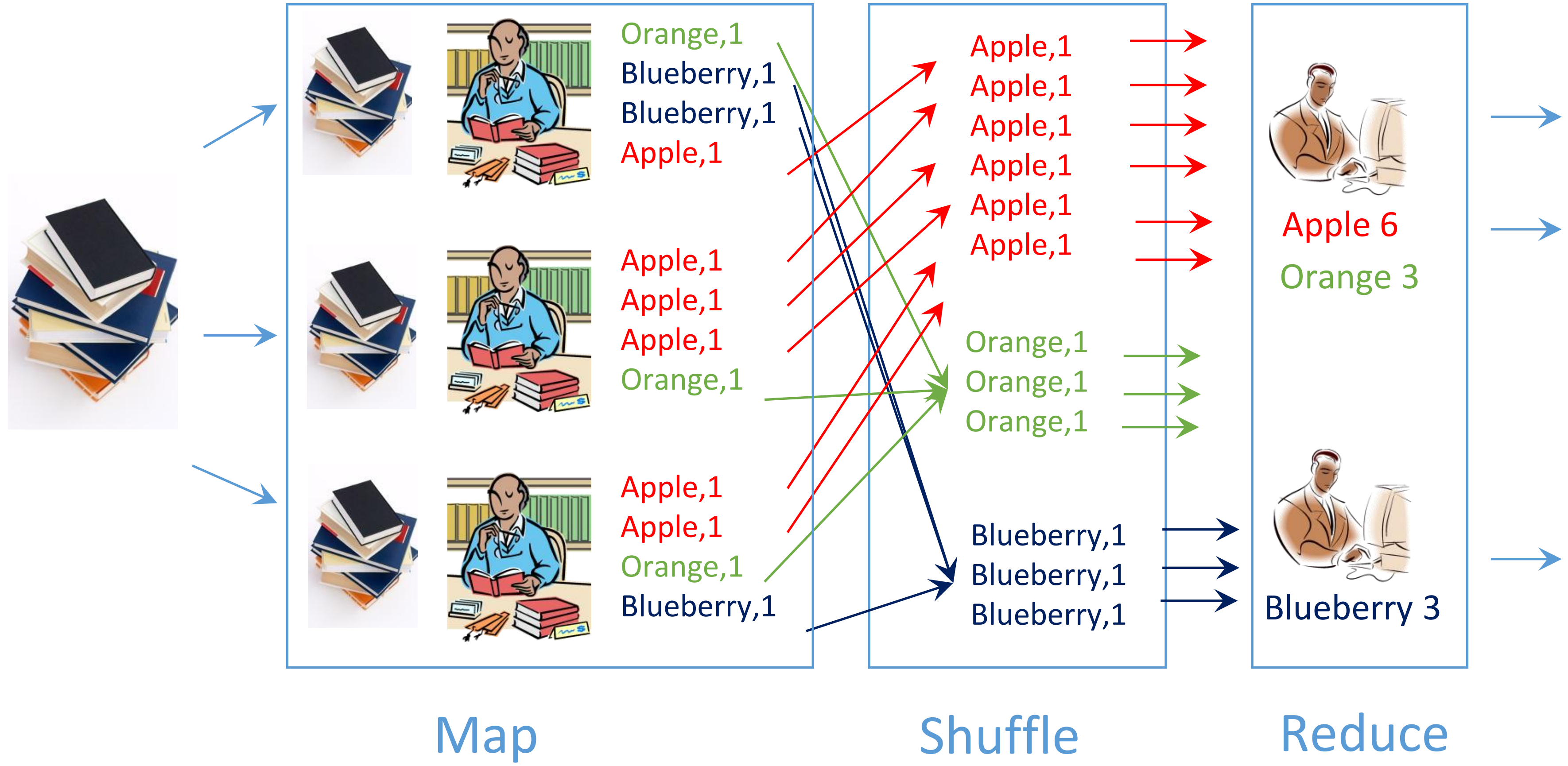
MapReduce Example

What if we want to count the number of times all fruits appeared in these books?

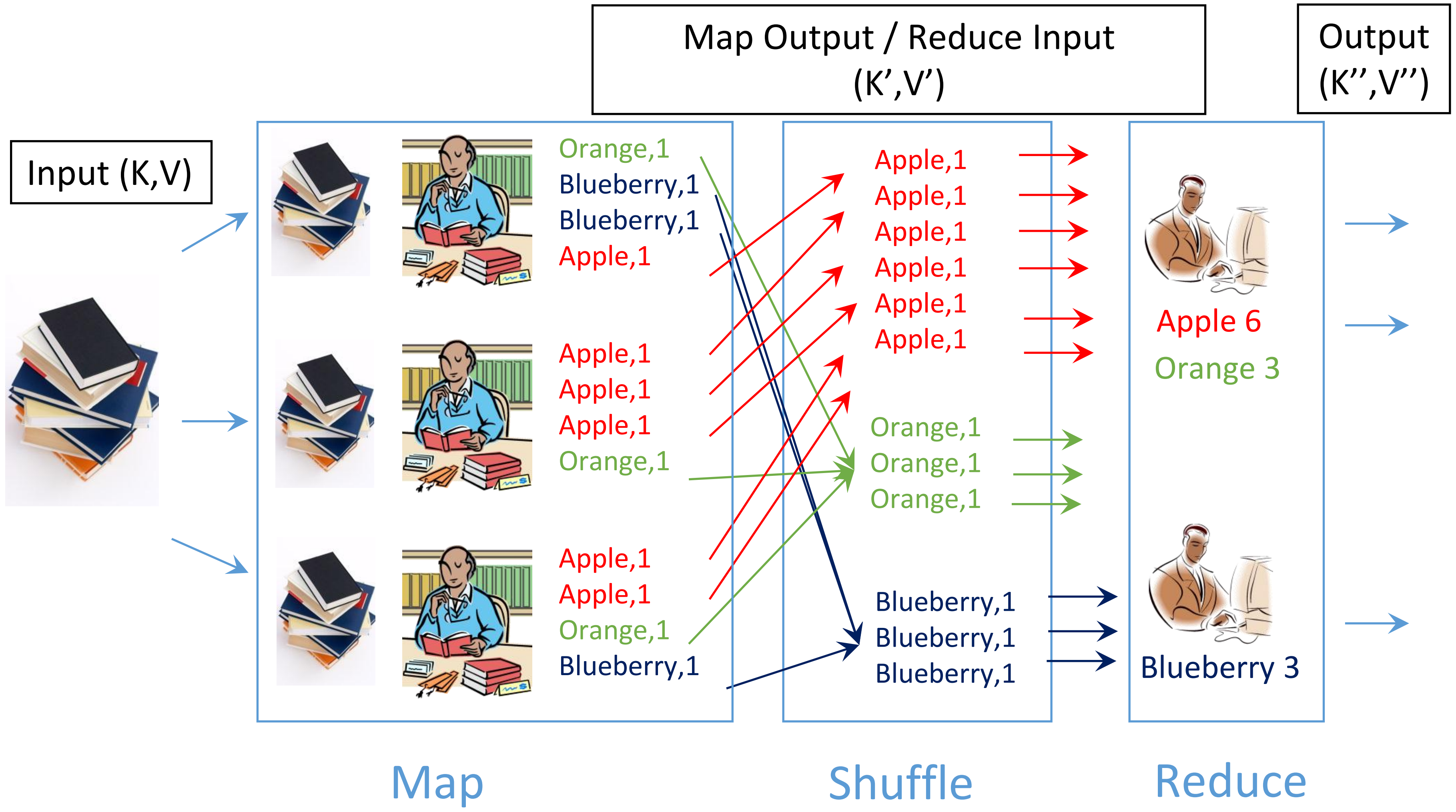


You can have multiple aggregators, each one working on a distinct set of “fruits”.

MapReduce Example



MapReduce Example



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
  - Sort and group by key:
    - Split keys and assign them to reducers (based on hashing)
    - Each key will be assigned to exactly one reducer
- Reduce
  - Input: mapper’s output (key-value pairs)
  - Each reducer will work on one or more keys
  - Output: the result needed
- Produce final output
  - Collect all output from reducers
  - Sort them by key

Mapreduce: Framework

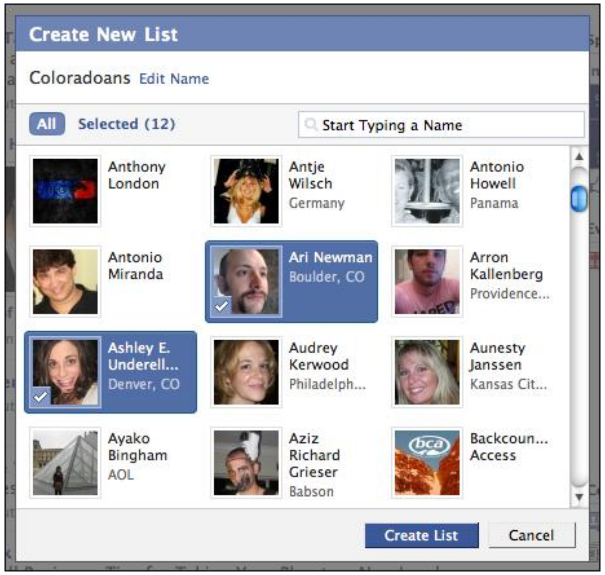
- The MapReduce framework takes care of:
  - Partitioning the input data
  - Scheduling the program’s execution across a set of machines
  - Perform the group by key (sort & shuffle) step
  - Handling machine failures
  - Manage required inter-machine communication

Parallelism in MapReduce

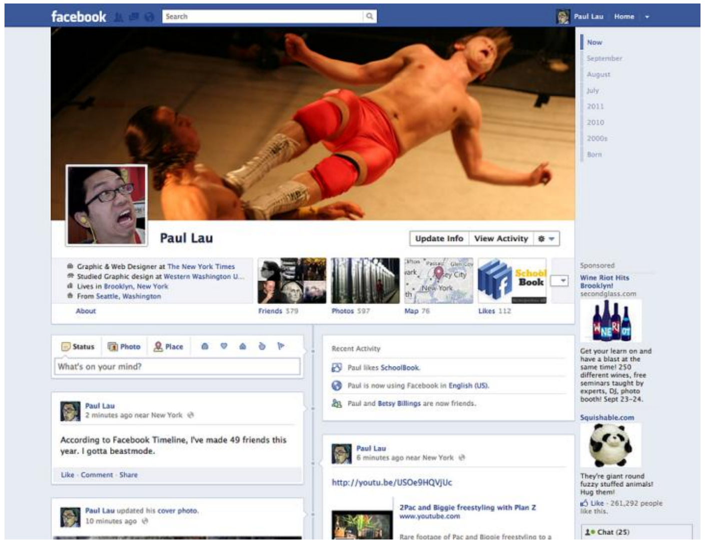
- Mappers run in parallel, creating different intermediate values from input data
- Reducers also run in parallel, each working on different keys
- However, reducers cannot start until all mappers finish
  - The Shuffle can start early as soon as the intermediate data from the mappers is ready

Example: Friend/Product Suggestion

- Facebook gathers information on your profile and timeline
  - e.g. contact list, messages, direct comments made, page visits, common friends, workplace/residence nearness
  - This info is dumped into a log or a database



Anthony London
Antje Wilsch Germany
Antonio Howell Panama
Antonio Miranda
Ari Newman Boulder, CO
Arron Kallenberg Providence...
Ashley E. Underell... Denver, CO
Audrey Kerwood Philadelph...
Aunesty Janssen Kansas Cit...
Ayako Bingham AOL
Aziz Richard Grieser Babson
Backcoun... Access



Paul Lau

Graphic & Web Designer at The New York Times
Studied Graphic design at Western Washington U...
Lives in Brooklyn, New York
From Seattle, Washington

Paul likes SchoolBook.
Paul is now using Facebook in English (US).
Paul and Betsy Billings are now friends.

Paul Lau
2 minutes ago near New York
According to Facebook Timeline, I've made 49 friends this year. I gotta beastmode.

Paul Lau
6 minutes ago near New York
http://youtu.be/USOe9HQVjUc
2Pac and Biggie freestyling with Plan Z
www.youtube.com
Rare footage of Pac and Biggie freestyling to a

Paul Lau updated his cover photo.
10 minutes ago

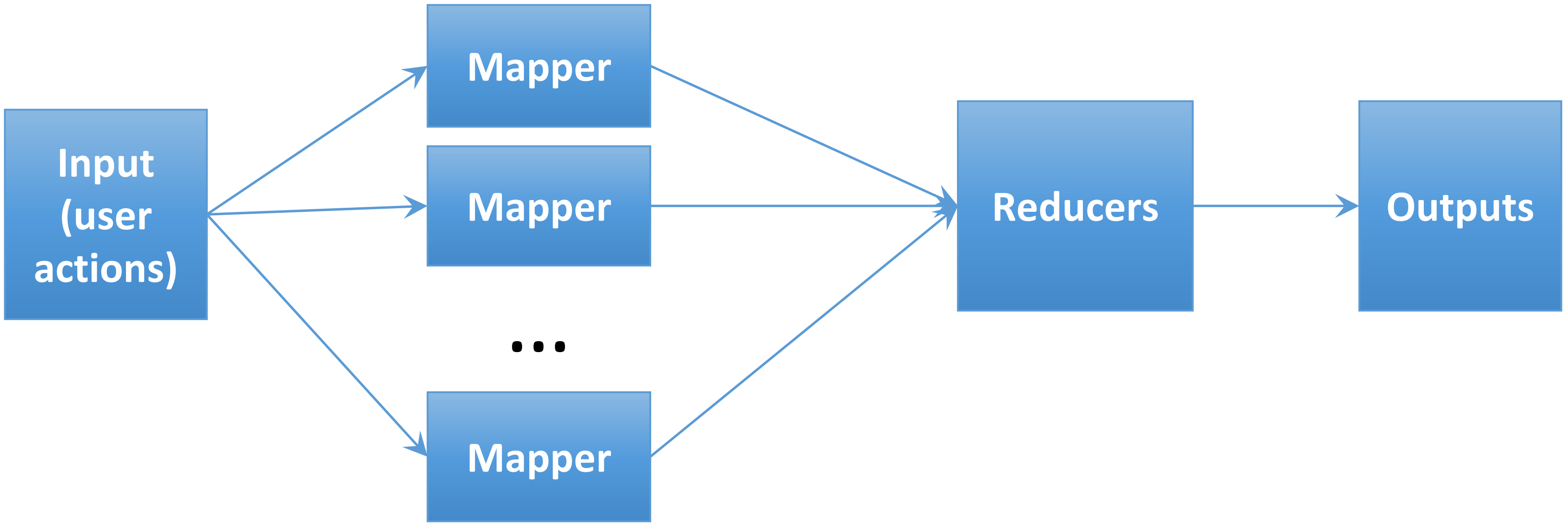
Wine Riot Hits Brooklyn!
secondglass.com
Get your learn on and have a blast at the same time! 250 different wines, free seminars taught by experts, DJ, photo booth! Sept 23-24.

Squishable.com
They're giant round fuzzy stuffed animals! Hug them!
Like · 261,292 people like this.

Real Example: Friend/Product Suggestion

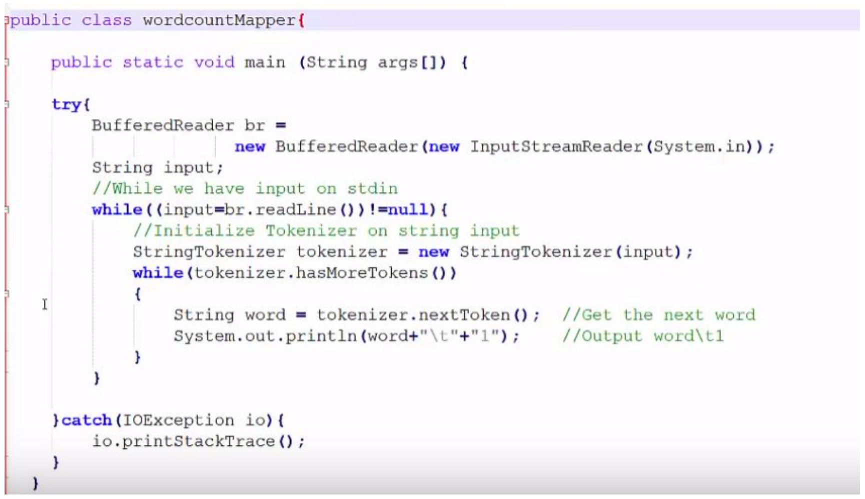
Generate key-value pairs:
Key: Friends pair
Value: Friends statistics (e.g. common friends)
e.g. (Tom, Sara statistics)

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

- Setup a **Streaming Elastic MapReduce** job flow in AWS
- Write simple Mapper and Reducer in the language of your choice
- Example Job Flow: Wordcount provided in writeup

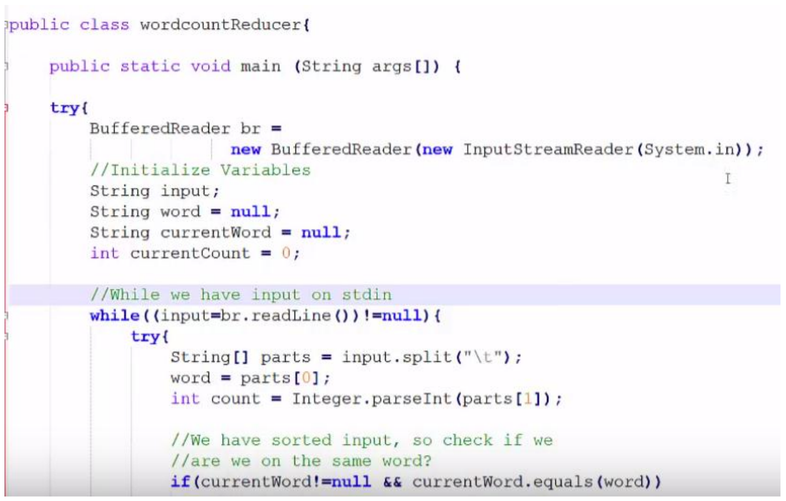


```
public class wordcountMapper{

    public static void main (String args[]) {

    try{
        BufferedReader br =
                    new BufferedReader(new InputStreamReader(System.in));
        String input;
        //While we have input on stdin
        while((input=br.readLine())!=null){
            //Initialize Tokenizer on string input
            StringTokenizer tokenizer = new StringTokenizer(input);
            while(tokenizer.hasMoreTokens())
            {
                String word = tokenizer.nextToken();  //Get the next word
                System.out.println(word+"\t"+"1");    //Output word\t1
            }
        }

    }catch(IOException io){
        io.printStackTrace();
    }
  }
```



```
public class wordcountReducer{

    public static void main (String args[]) {

    try{
        BufferedReader br =
                    new BufferedReader(new InputStreamReader(System.in));
        //Initialize Variables
        String input;
        String word = null;
        String currentWord = null;
        int currentCount = 0;

        //While we have input on stdin
        while((input=br.readLine())!=null){
            try{
                String[] parts = input.split("\t");
                word = parts[0];
                int count = Integer.parseInt(parts[1]);

                //We have sorted input, so check if we
                //are we on the same word?
                if(currentWord!=null && currentWord.equals(word))
```

How to write the Mappers and Reducers?

- The programs must read input files through stdin
- They have to write output through stdout
- Mapper, reducer and input data should be in S3
- Test your program on a local machine before launching a cluster!

```
cat input | mapper | sort |reducer > output
```

- Launch a cluster to process the data (Budget: $15)

How to Work on a Budget

- You will need to create an EMR cluster
  - EMR has additional hourly cost per instance.
  - Example: 10 x m1.large = 10 x (0.175 + 0.044) = **$2.19 per hour!**
  - Total time you have: ~ 6.84 hours in this configuration
- **Spot Instances are your friend**:
  - Same cluster @ spot pricing = 10 x (0.0161 + 0.044) = **$0.601 per hour!**

P1.2 Grading

- P1.2 is 9% of your grade for the course!
  - Easy to do well and easy to lose a lot of points!
- For this checkpoint, assign the tag with
  - Key: **Project** and Value: **1.2** for all resources
  - Before Launching!
  - No tags → 10% grade penalty
- Budget
  - For P1.2, each student’s budget is $15
  - Exceeding $15 → 10% project penalty
  - Exceeding $30 → 100% project penalty
- Plagiarism → the lowest penalty is 200% & potential dismissal

Upcoming Deadlines

- **Quiz 2**
  - Deadline, Friday, Sept 18, 11:59pm ET
- **Complete Project 1.2 (Using Elastic MapReduce)**
  - Deadline, Sunday, Sept 20, 11:59pm ET

Demo

- Quick Tour of AWS
  - EMR
  - On-Demand and Spot Instances
  - Billing and Monitoring Costs
- Demo: Wordcount on EMR
- Auto-grader for P1.2
  - How to make a submission