15-319 / 15-619 Cloud Computing

Recitation 3

Jan 31 and Feb 2, 2017

Overview

- **Administrative Issues**
- **Last Week's Reflection**
  - Project 1.1, OLI Unit 1, Quiz 1
- **This Week's Schedule**
  - Project 1.2, OLI Unit 2, Module 3 and 4, Quiz 2
- **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 (@10) 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
- Do not share anywhere (Piazza, etc…)
- 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

Reflecting on Last Week

- Reading:
  - Unit 1: Introduction to Cloud Computing
    - Modules 1 & 2
  - Quiz 1: Introduction to Cloud Computing
  -
- Project:
  - Project 1.1:
    - Wikipedia Dataset
    - Filtering one hour's worth of data

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
  - But if you can fit everything in-memory, big win
  - A better approach: work from disk, build a processing pipeline
  - Write programs that process the data line by line
- Common Issues
  - Encoding (UTF-8)
  - Assuming the size of the dataset

The early birds get the worm

Submitter	Last Submission Time
weid1@andrew.cmu.edu	2017-01-24 01:50:48
ndong1@andrew.cmu.edu	2017-01-24 02:00:14
pwang1@andrew.cmu.edu	2017-01-24 21:00:11
lewenz@andrew.cmu.edu	2017-01-24 21:24:07
lle1@andrew.cmu.edu	2017-01-24 22:39:19
cchao1@andrew.cmu.edu	2017-01-25 09:05:56
kunminl@andrew.cmu.edu	2017-01-25 14:39:10
jzhu2@andrew.cmu.edu	2017-01-25 18:22:45
jiupengs@andrew.cmu.edu	2017-01-25 19:04:22
anhongg@andrew.cmu.edu	2017-01-25 19:23:49
mplamann@andrew.cmu.edu	2017-01-25 22:03:27
jzhan1@andrew.cmu.edu	2017-01-25 23:58:26

This Week's Schedule

- Complete Unit 2 (Modules 3 & 4)
- **Quiz 2**
  - Deadline, Friday, 11:59pm ET
- **Complete Project 1.2**
  - Using MapReduce on EMR / HDInsight / Dataproc
  - Deadline, Sunday, 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

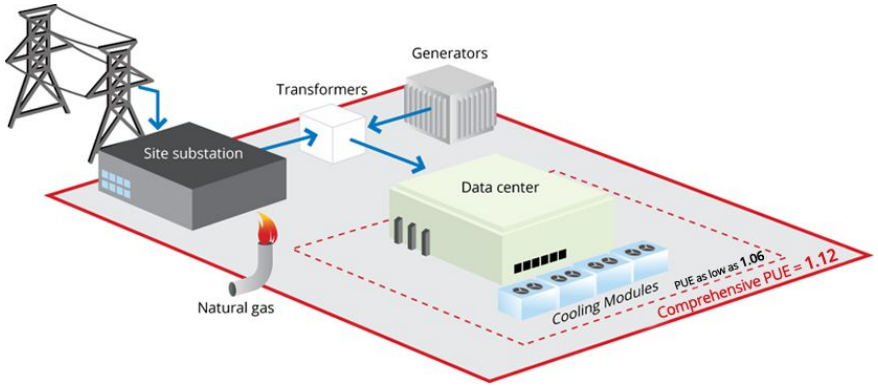
Module 3: Data Center Trends

- 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
  - 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/

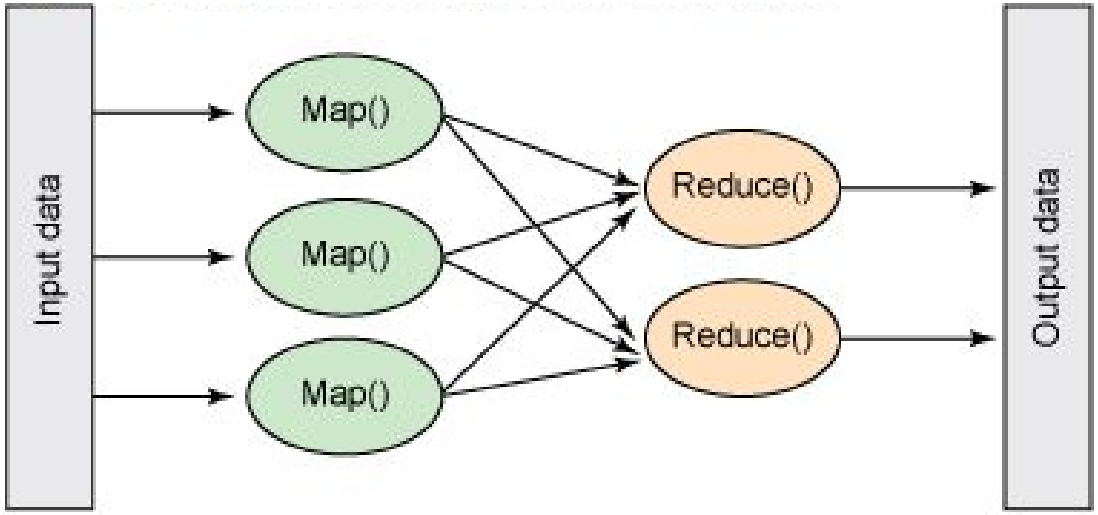
Project 1.2

- 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?

- 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

Vampire,1

Vampire,1
Vampire,1
Vampire,1

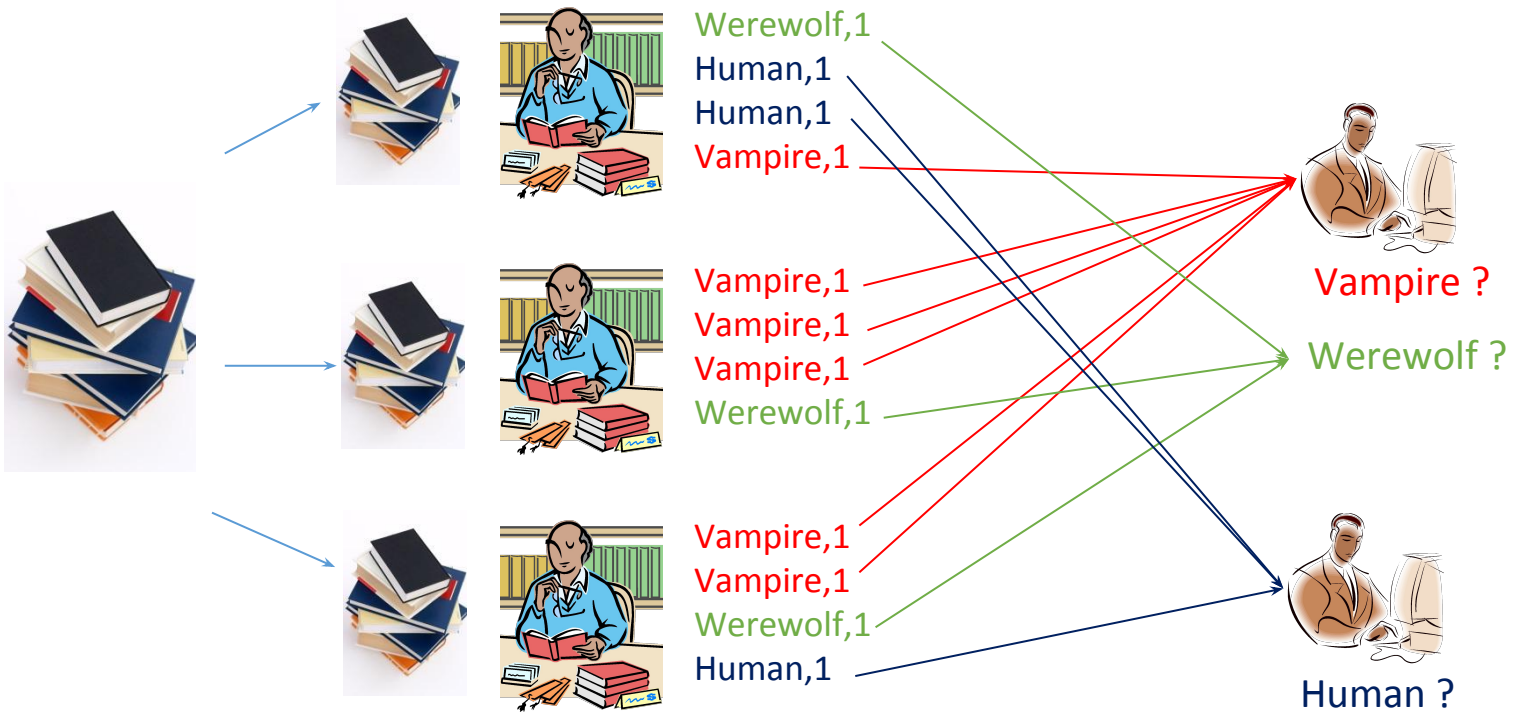
Vampire,1
Vampire,1

How many times does the word “vampire” appear in all “books” in The Twilight Saga?

I heard 6 “Vampire”s !

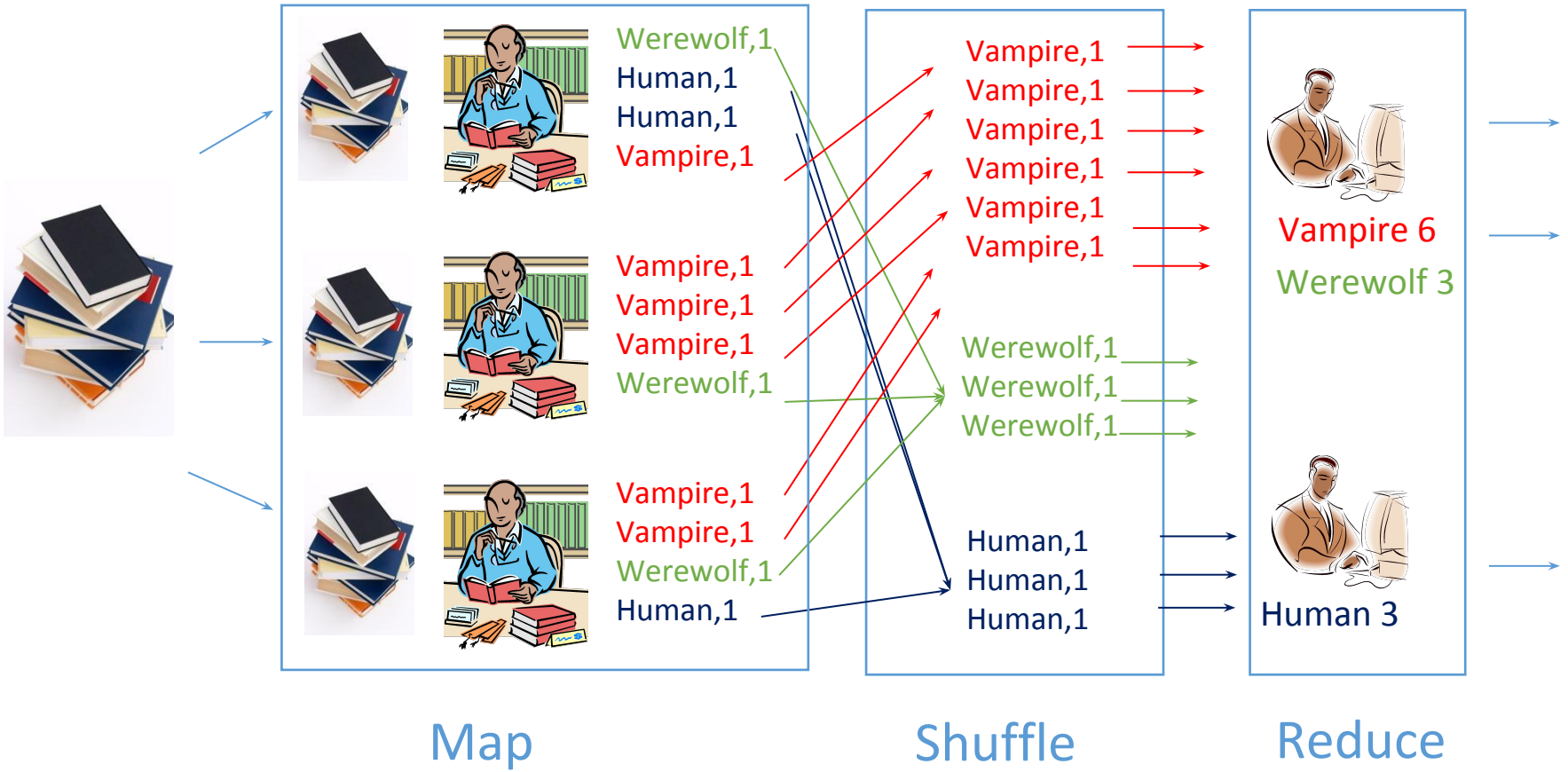
MapReduce Example

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

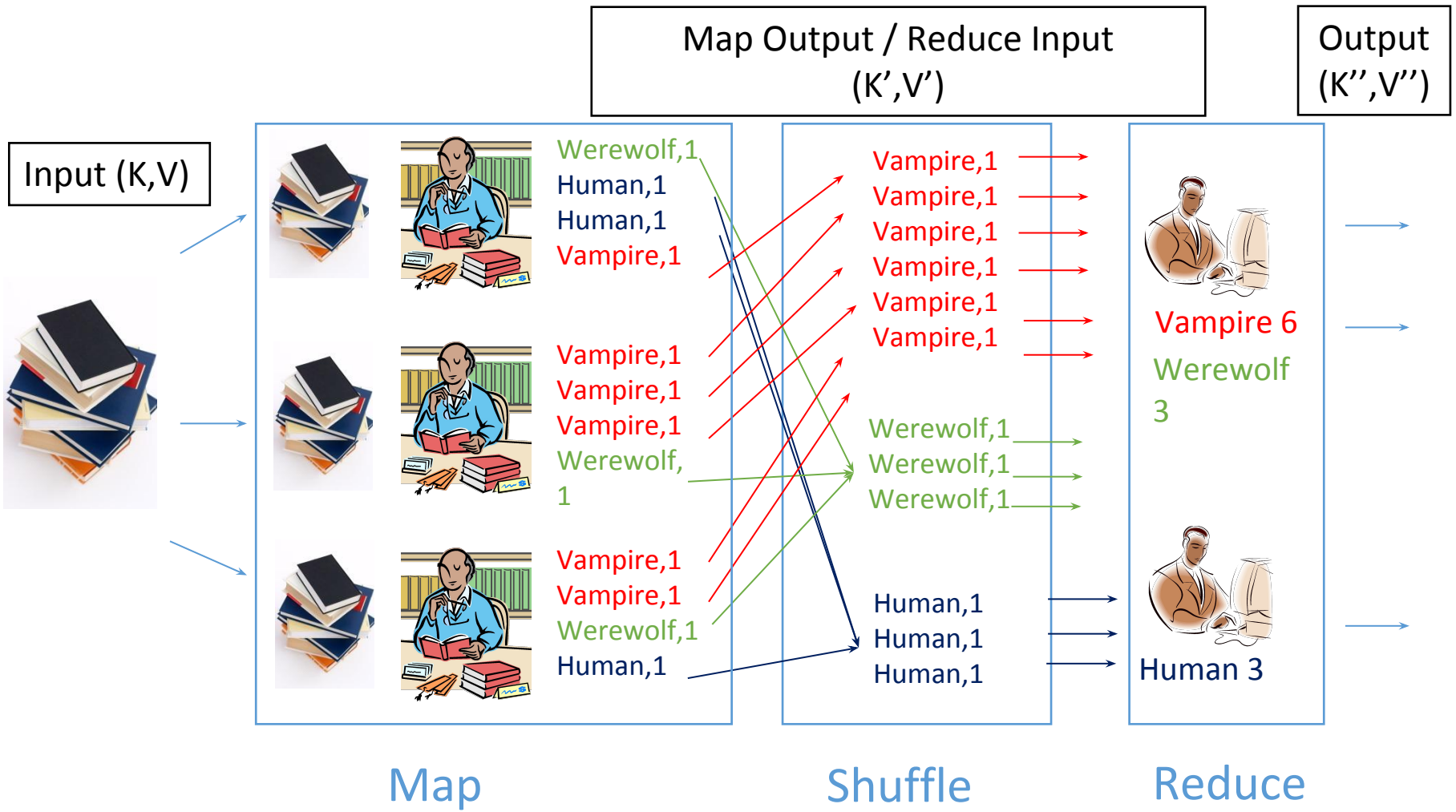


You can have multiple aggregators, each one working on a distinct set of species.

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. (Vampire, 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

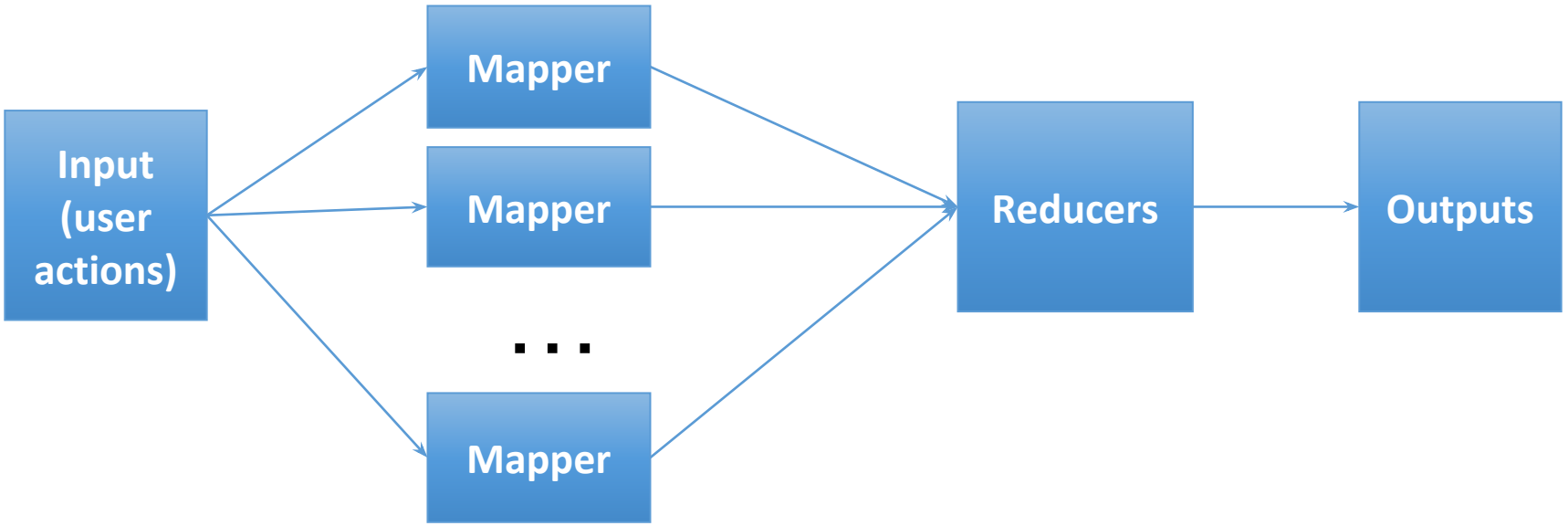
Example: Friend/Product Suggestion

- 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

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
  - AWS EMR or Azure HDInsight
- 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 or a Storage account
- 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 m3.xlarge = 10 x (0.266 + 0.070) = **$3.36 per hour!**
  - Total time you have: ~ 4.46 hours in this configuration

- **Spot Instances are your friend**:
  - Same cluster @ spot pricing = 10 x (0.0445 + 0.044) = **$0.885 per hour!**
  - Total time you have: ~ 16.95 hours in this configuration

P1.2 Logistics

- For this checkpoint, use tags with
  - Key: **Project** and Value: **1.2** for all resources
  - Tag before Launching! And check after 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

P1.2 Program Flow - AWS EMR

- Specify given S3 location as input
- At the mapper:
  - Read all data as lines from stdin
  - Find the filename associated with each line in the mapper
    - Use mapreduce_map_input_file to get filename
    - Do not use the filename to open() the file
  - Extract the date and the title, view count
- At the reducer:
  - Aggregate daily counts and print the ones over the threshold

P1.2 Program Flow - Azure HDInsight

- Specify given Azure Storage (wasb://) location as input
- Mapper and Reducers are the same as in EMR
- You will have to log into the HDInsight cluster and run the job with Yarn.
- The Hadoop output (part-00000) will be stored in the clusters Storage account.

P1.2 Program Flow - GCP Dataproc

1. Specify given Google Storage (gs://) location as input (the correct location is specified in the writeup).

2. Once again, use the same Mappers and Reducers that you invoked for EMR and HDInsight.

3. You will have to store your code and place it on the master node of your Dataproc cluster and run it with yarn (hint: use Google Storage).

4. The output will be stored in a Google Storage location of your own creation (the same place you chose store your code in step 3).

This Week's Deadlines

- Complete Unit 2 (Modules 3 & 4)
- **Quiz 2**
  - Deadline, Friday, 11:59pm ET
- **Complete Project 1.2**
  - Using MapReduce on EMR / HDInsight / Dataproc
  - Deadline, Sunday, 11:59pm ET