15-319 / 15-619 Cloud Computing

Recitation 12

April 7th and 9th , 2015

Overview

- **Last week's reflection**
  - Project 3.5
- **Budget issues**
  - Tagging, 15619Project
- **This week's schedule**
  - Unit 5 - Modules 17
  - Project 4.1
- **Demo**
- **Twitter Analytics: The 15619Project**

Reflections on P3.5

- Implement a key-value store with different levels of consistency
  - Strong consistency
  - Causal consistency
- FAQs
  - Failure in consistency checker test
    - Best way: log requests and analyze
    - The checker also checks for response time of certain requests in some tests

Budget Issues

- Tag all resources(Piazza @1656)
  - Amazon EBS General Purpose SSD
  - Amazon EBS Provisioned IOPS
  - Amazon EBS Snapshot
- Untagged resources will be counted towards your weekly project
  - Keep a buffer

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
    - Hadoop 1.0
    - Hadoop 2.0 - YARN
  - Module 18: Distributed Analytics Engines for the Cloud: Spark
  - Module 19: Distributed Analytics Engines for the Cloud: GraphLab

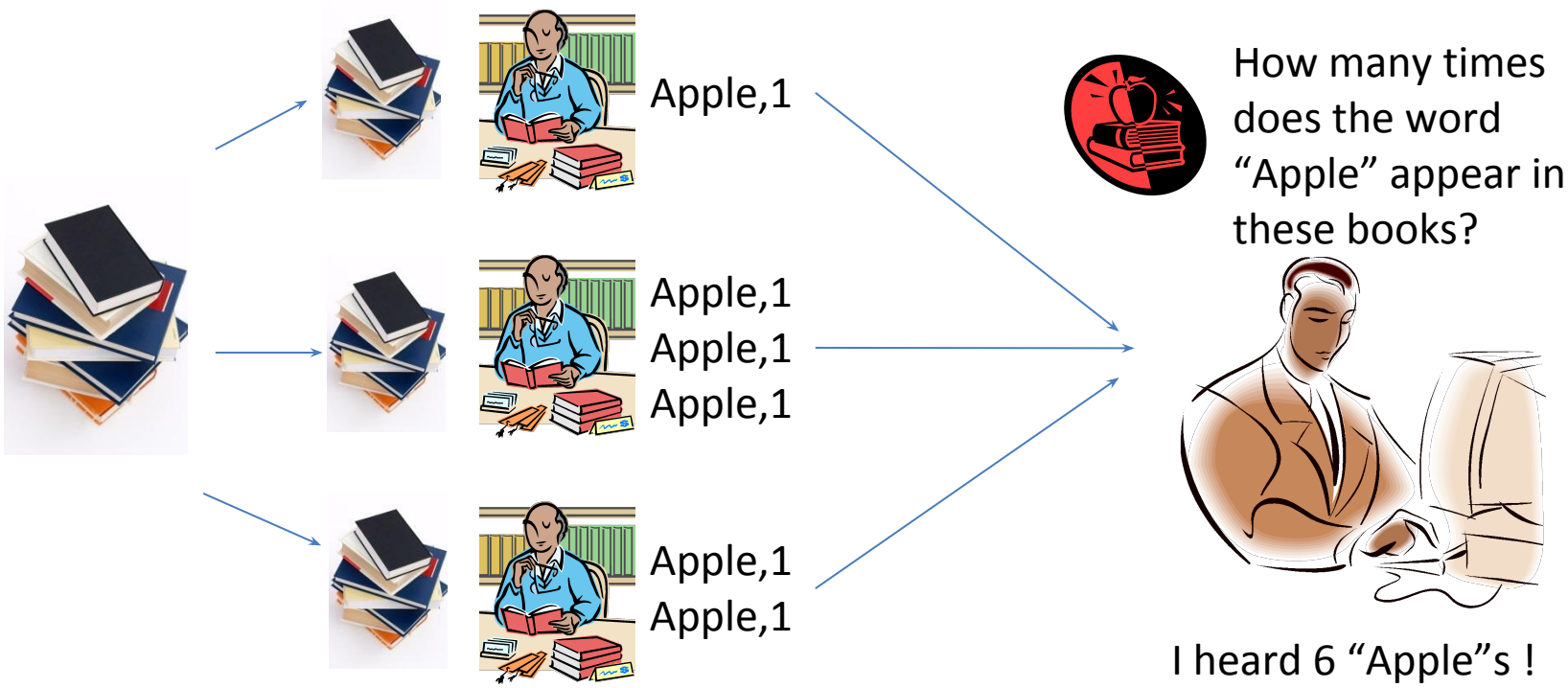
Project 4

- Project 4.1
  - MapReduce Programming Using YARN

- Project 4.2
  - Iterative Programming Using Apache Spark
- Project 4.3
  - Graph Programming Using GraphLab

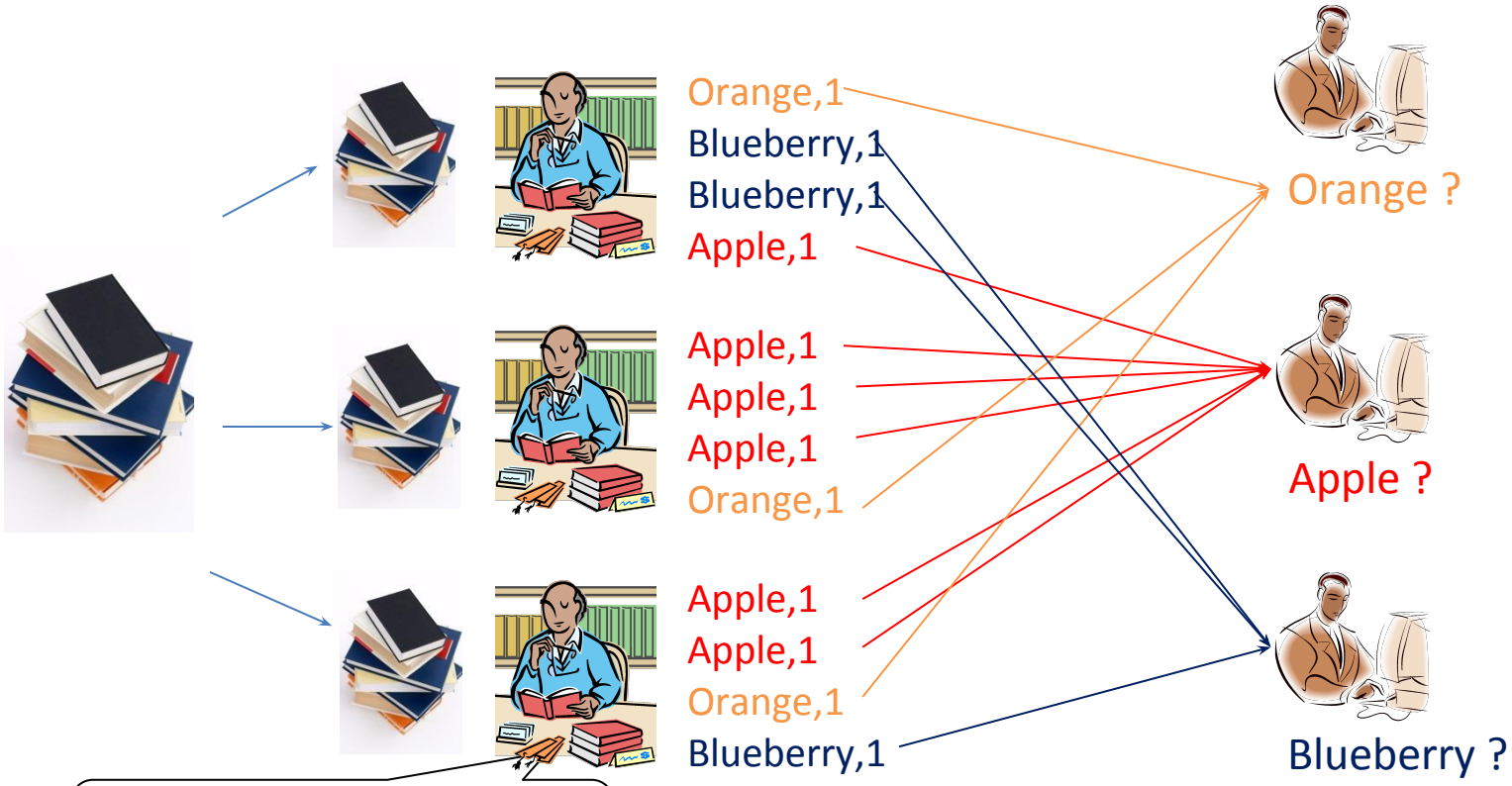
MapReduce - Introduced in Project 1

- The idea of MapReduce



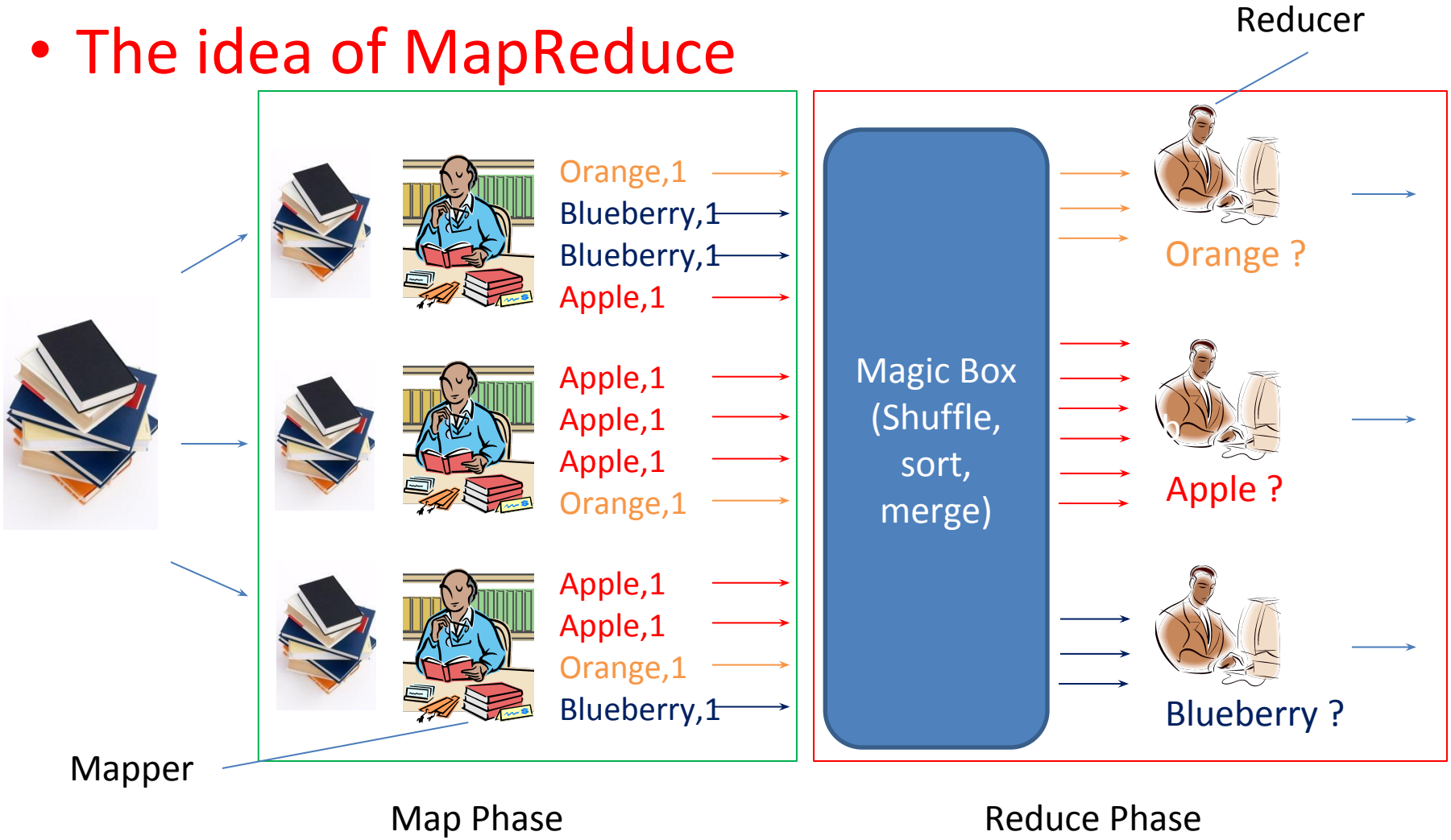
MapReduce - Overview

- The idea of MapReduce



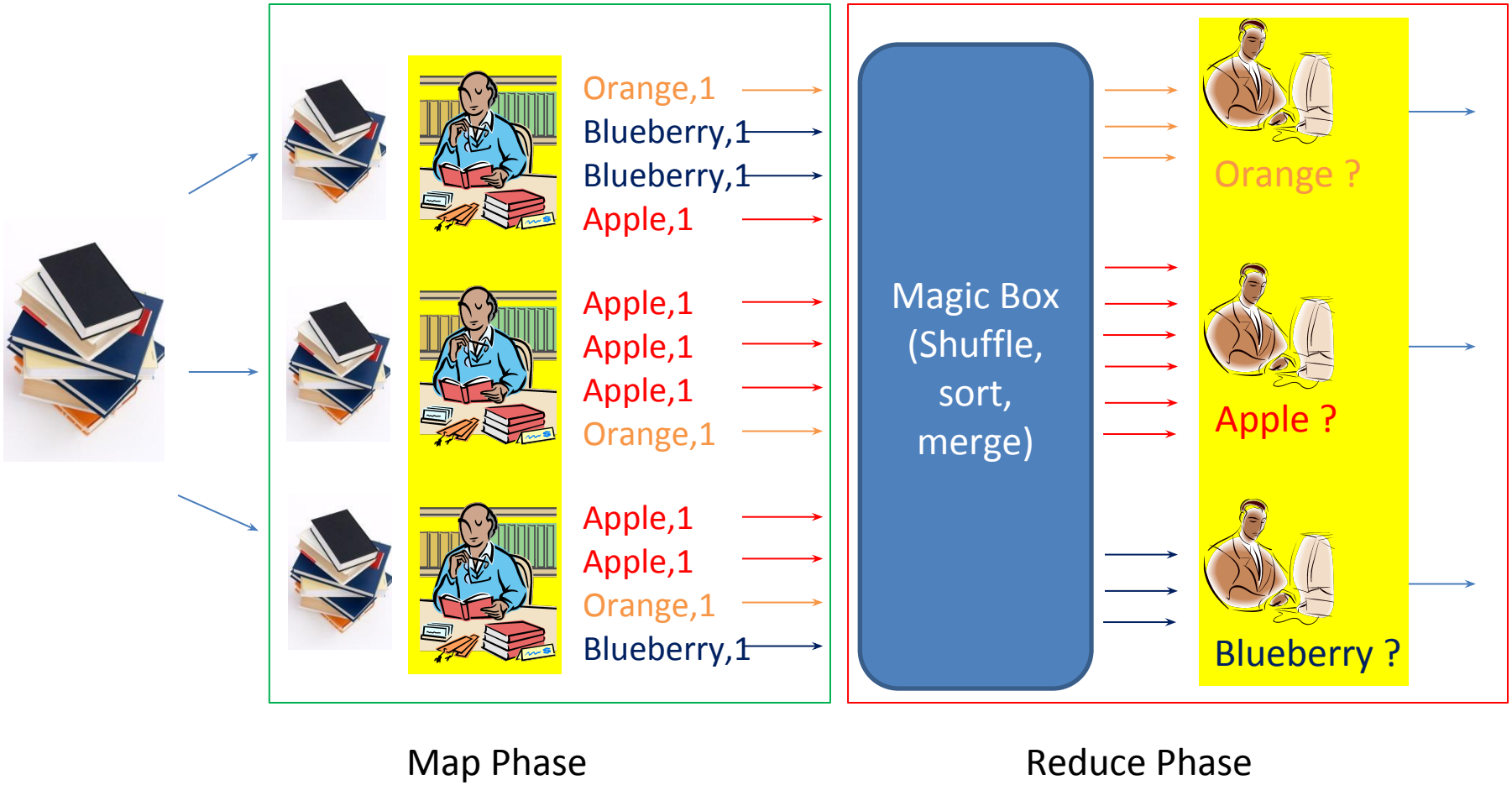
MapReduce - Phases

- The idea of MapReduce



MapReduce - This Week

- The idea of MapReduce



MapReduce - Data Types

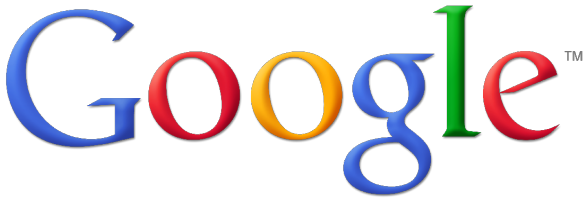
- Mapper (default)
  - Input: **key-value pairs**
    - **Key:** byte offset of the line
    - **Value:** the text content of the line
  - Output: **key-value pairs**
    - **Key:** specified by your program
    - **Value:** specified by your program based on what content you expect the reducer to receive as a list

(k1,v1) -> Mapper -> (k2,v2)

MapReduce - Data Types

- Reducer
  - Input: **key-value pairs**
    - A list of values for each key output from the mapper
  - Output: **key-value pairs**
    - The desired result from your aggregation

(k2,list(v2)) -> Reducer -> (k3,v3)



Proprietary		Open Source
GFS	→	HDFS
MapReduce	→	MapReduce
BigTable	→	HBase

Hadoop

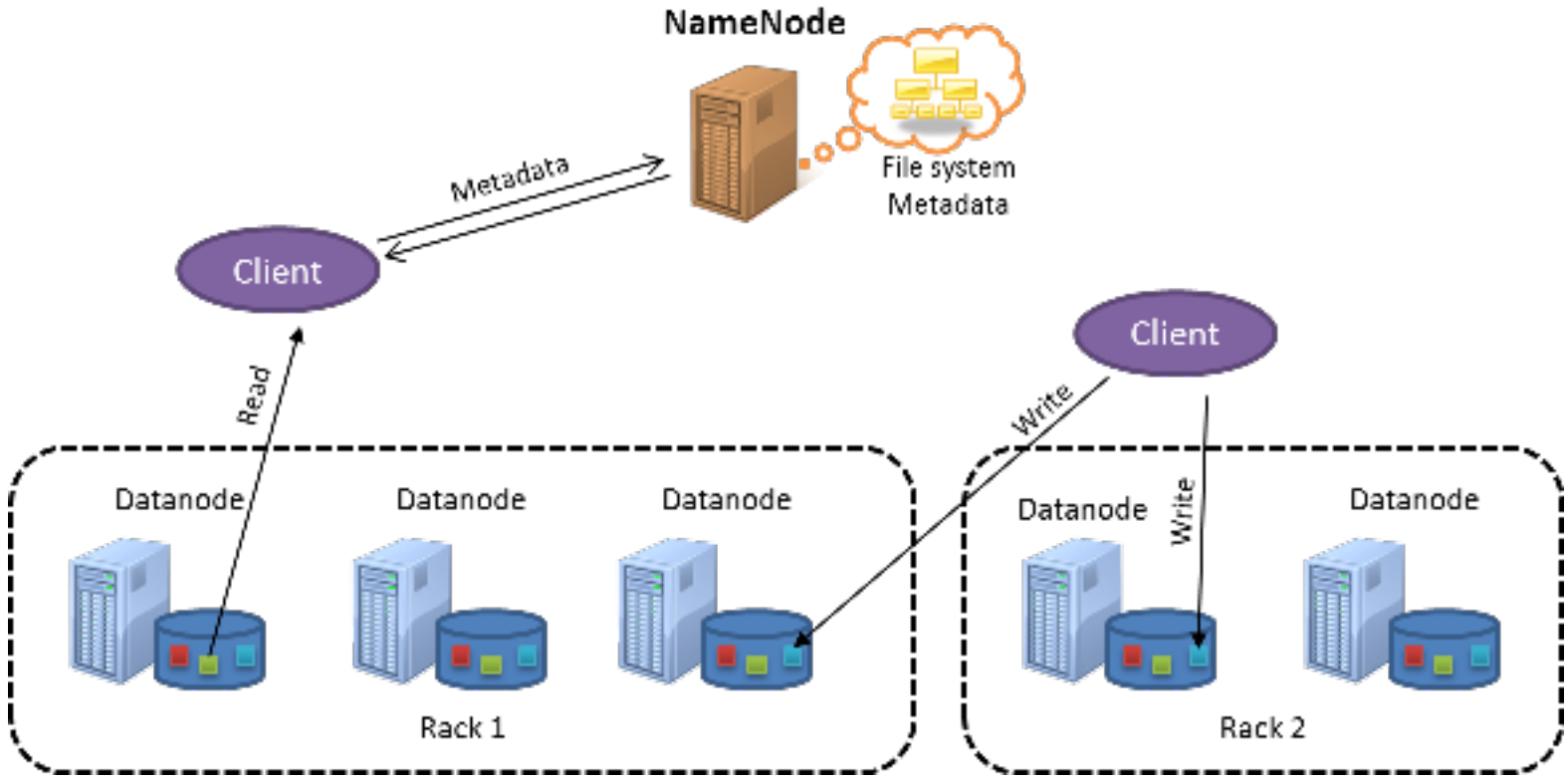
- MapReduce
  - A programming model for processing large data sets using a parallel distributed algorithm
- Apache Hadoop
  - A framework for running applications on a large cluster of commodity hardware
  - Implements the MapReduce computational paradigm
  - Uses HDFS for data storage
  - Engineers with little knowledge of distributed computing can write the code in a short period

HDFS - Distributed File System

- Paper
  - The Hadoop Distributed File System, Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, Yahoo!, 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)
- Purpose
  - Serve as the distributed storage to run Hadoop's MapReduce applications
  - An open-source framework which can be used by different clients with different needs

HDFS - Distributed File System

- Hadoop Distributed File System
- Open source version of Google File System

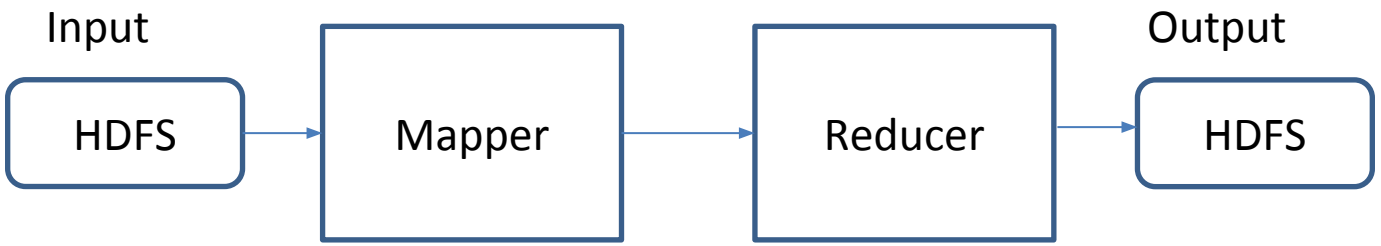


Using a Custom Jar in P4.1

- What is a custom JAR
  - Customize your java MapReduce program
  - Run the MapReduce JAR in EMR
- Why custom JAR
  - More resources: HDFS/HBASE/S3
  - More job configuration flexibility
  - More control of how the resources are utilized

Typical MapReduce Job

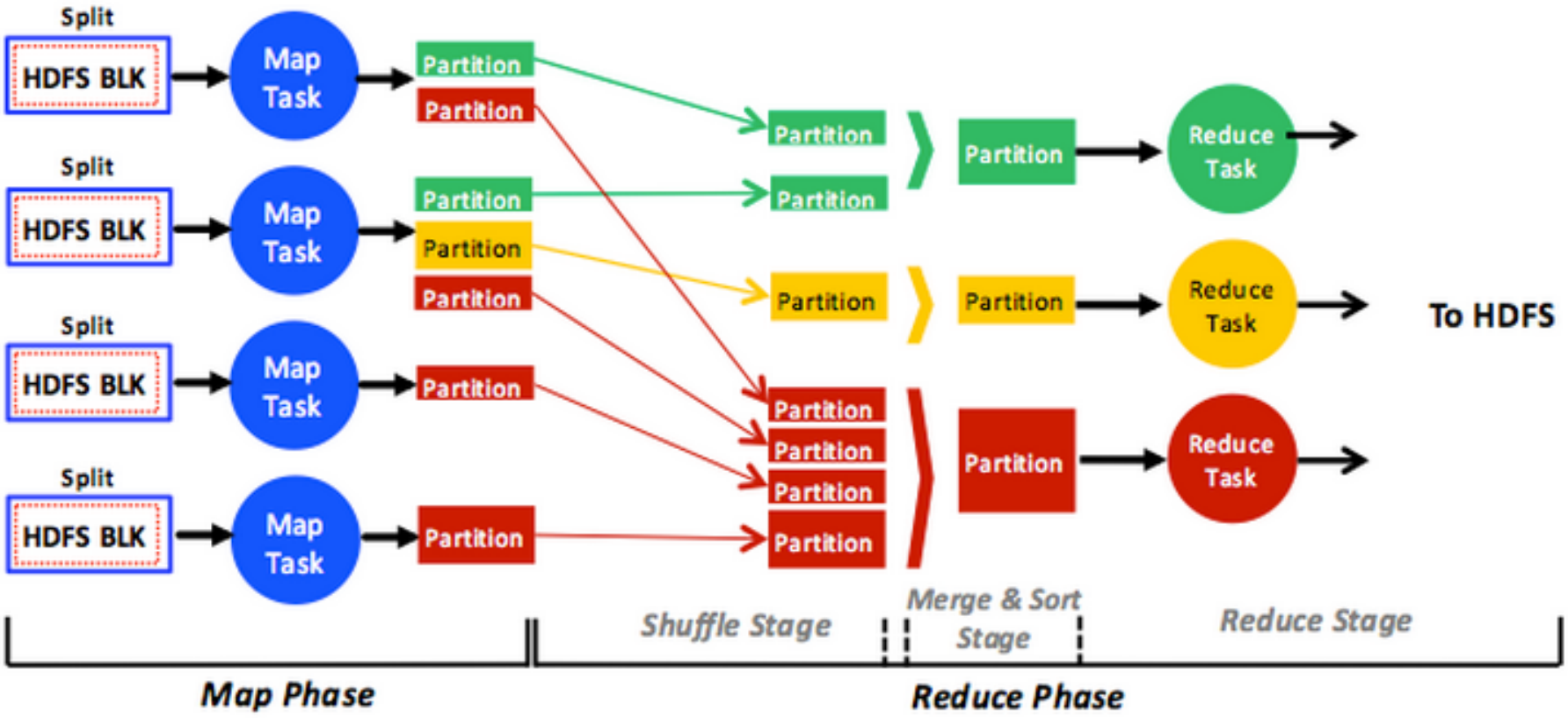
- Simplistic view of a MapReduce job



- You simply write code for the
  - Mapper
  - Reducer

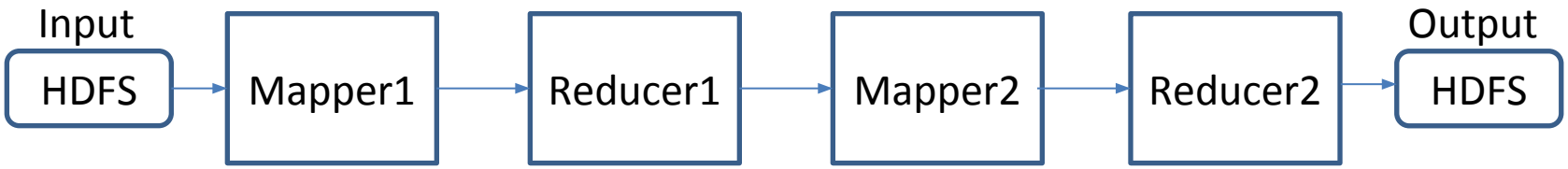
MapReduce and HDFS

- Detailed workflow

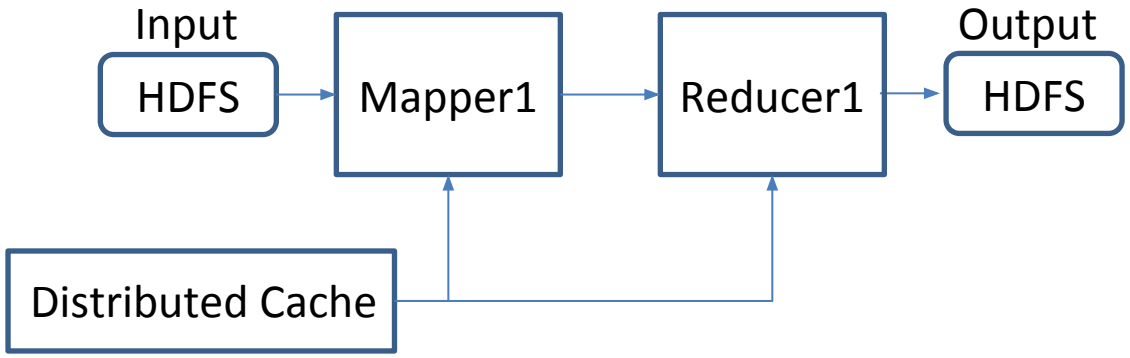


Cool things with MapReduce

- Chain of two MapReduce jobs



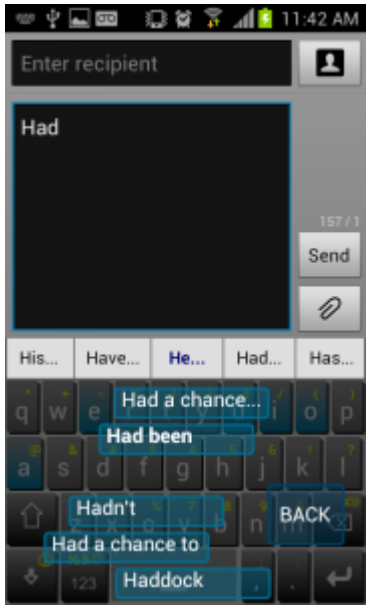
- Load external data into your program

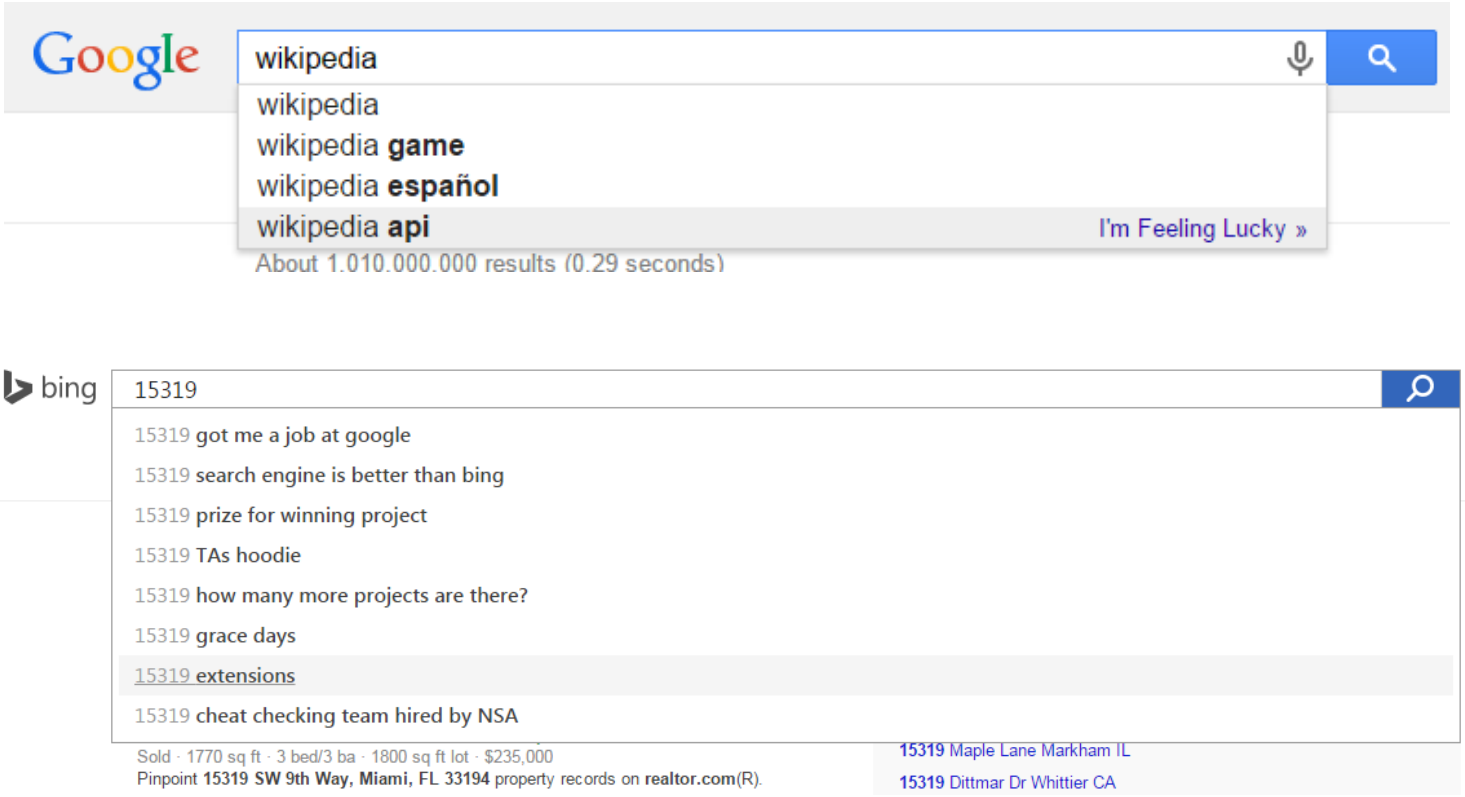


- Modify the behavior of FileInputSplit

Project 4.1 - Input Text Predictor

- Suggest words based on letters already typed





Project 4.1

- Input Text Predictor
  - Input Data
  - N-Gram Model
  - Statistical Language Model
  - Predict the next word given a phrase
- Have to use EMR Custom JAR
  - **CANNOT use EMR Streaming**

Construct an Input Text Predictor

1. Given a language corpus
   - Project Gutenberg (2.5 GB)
2. Construct an n-gram model of the corpus
   - An n-gram is a phrase with n contiguous words
   - For example a set of 1,2,3,4,5-grams with counts:
     - this 1000
     - this is 500
     - this is a 125
     - this is a cloud 60
     - this is a cloud computing 20

Construct an Input Text Predictor - 2

3. Build a statistical language model to calculate the probability of a word appearing after a phrase

$$\Pr(\text{word} \mid \text{phrase}) = \frac{\text{Count}(\text{phrase} + \text{word})}{\text{Count}(\text{phrase})}$$

$$\Pr(\text{is} \mid \text{this}) = \frac{\text{Count}(\text{this is})}{\text{Count}(\text{this})} = \frac{500}{1000} = 0.5$$

$$\Pr(a \mid \text{this is}) = \frac{\text{Count}(\text{this is } a)}{\text{Count}(\text{this is})} = \frac{125}{500} = 0.25$$

4. Load data to HBase and predict the next word based on the probabilities

Generate n-gram

- An *n*-gram is a phrase with *n* contiguous words

Example Phrase: This is interesting because this is a cloud computing course

#	1-gram	Count
1	this	2
2	is	2
3	interesting	1
4	because	1
5	a	1
6	cloud	1
7	computing	1
8	course	1

#	2-gram	Count
1	this is	2
2	is interesting	1
3	interesting because	1
4	because this	1
5	is a	1
6	a cloud	1
7	cloud computing	1
8	computing course	1

#	3-gram	Count
1	this is interesting	1
2	is interesting because	1
3	interesting because this	1
4	because this is	1
5	this is a	1
6	is a cloud	1
7	a cloud computing	1
8	cloud computing course	1

#	4-gram	Count
1	this is interesting because	1
2	is interesting because this	1
3	interesting because this is	1
4	because this is a	1
5	this is a cloud	1
6	is a cloud computing	1
7	a cloud computing course	1
8		

#	5-gram	Count
1	this is interesting because this	1
2	is interesting because this is	1
3	interesting because this is a	1
4	because this is a cloud	1
5	this is a cloud computing	1
6	is a cloud computing course	1
7		
8		

#	6-gram	Count
1	this is interesting because this is	1
2	is interesting because this is a	1
3	interesting because this is a cloud	1
4	because this is a cloud computing	1
5	this is a cloud computing course	1
6		
7		
8		

Statistical Language Model

- Provide a mechanism to solve common natural language processing problems
- Examples: speech recognition, machine translation and intelligent input method
- SLM estimates the probability of a word given the previous phrase
- N-gram model is one of the most popular mechanisms to generate an SLM today

Statistical Language Model

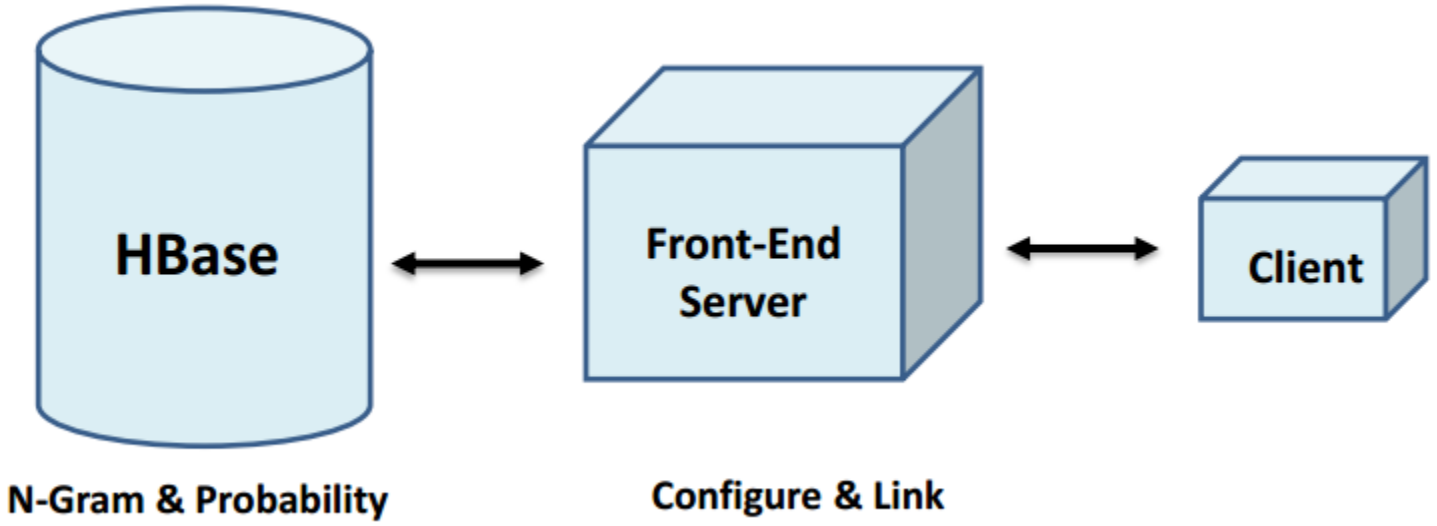
- Build a statistical language model that calculates the probability of a word appearing after a phrase

Options	Count	Probability
this was	150	0.15
this is	500	0.50
this day	250	0.25
this kiss	25	0.03
this boy	75	0.08

Options	Count	Probability
this is	500	0.50
this day	250	0.25
this was	150	0.15
this boy	75	0.08
this kiss	25	0.03

Load and Predict

- Load data into HBase
- Connect HBase with the PHP-based front end server to provide a functional web service.



Recommendations

- Test for correctness with small datasets first
- **Don't** start a new cluster for every job
  - EMR will charge you one hour of usage for instances even though your EMR job failed to start
- Optimize your code to accelerate MapReduce before seeking other optimization methods
  - Pay attention to your code efficiency
- Version of Hadoop
  - should match the version shown in EMR AMI
- Start early

Upcoming Deadlines

- P4.1 MapReduce Programming Using YARN
  - Due: 11:59PM ET April 12th (Sunday)
- 15619Project Phase3
  - Deadline: 18:59PM ET April 15th (Wednesday)
    - Live Test, due 18:59PM ET Apr 15th

Project 4.1

- Demo

Overview

- Run 2 MapReduce jobs to generate a language model
- First step: generate n-grams
- Second step: generate language model
- Third step: connect user interface to HBase

Grading

- Use submitter file to autograde your answers
- Ngrams
- Run the command *./submitter -n* with the top 100 ngrams in a file called “ngrams”.

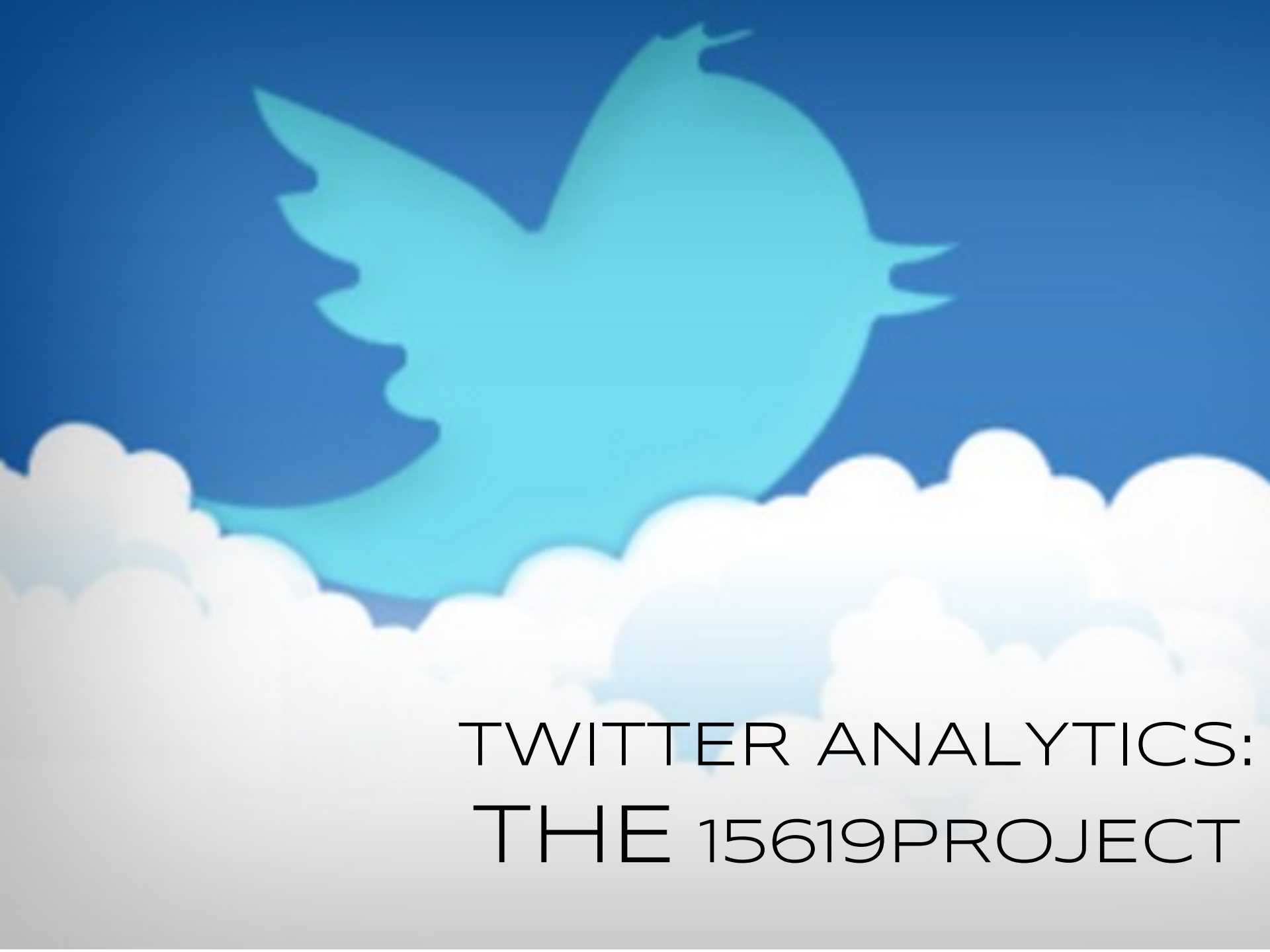
Grading

- Model
- Run the command *./submitter -m* to autograde you language model and your interface
- Code files for ngram and model will be manually graded

Hive Shell for ad-hoc queries

- Run SQL like queries over distributed storage(HDFS/S3)
- SELECT, WHERE, ORDER BY,...
- https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Select

Questions?



TWITTER ANALYTICS:
THE 15619PROJECT

Phase 2 Live Test

Congratulations UnusualItem!

MySQL

Team	Phase Score
UnusualItem	203
ComeATeamName	186
HYPER	160
CloudWalker	149
z2m	99
T.K.Lim	95
Lays	92
Oak	89
Penguins	89
YouKnowWho	86
etc	75

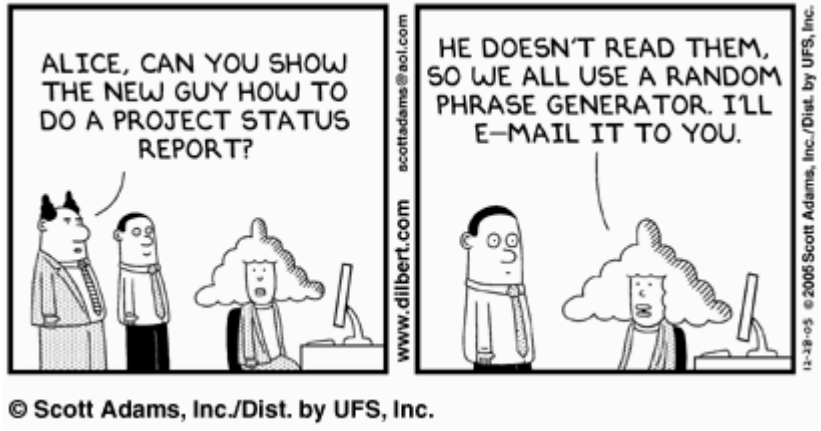
HBase

Team	Phase Score
UnusualItem	90
T.K.Lim	82
YouKnowWho	68
The Three Foes	57
Thunder & Lightning	57
ComeATeamName	49
Wegotateam	46
Oak	44
etc	43
Cloud007	41
9527	41

What's due soon?

- Phase 3 Deadline
  - **Submission of one URL by 18:59 ET (Pittsburgh) Wed 4/15**
    - **Live Test from 8 PM to midnight ET**
  - Choose one database
  - Can only use m1.large or cheaper t1, t2, m1, m3 instances
  - Fix Q1,Q2,Q3,Q4 if your Phase 2 did not go well
  - New queries Q5 and Q6.
  - Phase 3 counts for 60% of 15619Project grade

Phase 3 Report [VERY IMPORTANT]



- Start early
- Document your steps
- Identify and isolate the performance impact of each change you make
- Document your ideas and experiments

MAKE A QUANTITATIVE, DATA-DRIVEN REPORT

Query 5: Twitter Rankster

- Request: a list of userids and a date range
- You should award points to the users based on these rules:
  - +1 per unique tweet sent by the user in the time interval
  - +3 per friend (based on the maximum value of user.friends_count in the time interval)
  - +5 per follower (based on the maximum value of user.followers_count in the time interval)

Query 5: Twitter Rankster

```
GET /q5?userlist=12,14,16,18,20&start=2010-01-
01&end=2014-12-31


Team,1234-5678-1234,1234-5678-1234,1234-5678-1234
12,173
16,155
14,99
20,99
18,55
```

Query 6: Hermit Finder

- Request: A range of userids
- You should count the number of users where:
  - userid is between M and N inclusive,
  - has at least one tweet but none of his/her tweets contain location information.

```
GET /q6?m=0&n=9999999999
```

```
Team,1234-5678-1234,1234-5678-1234,1234-5678-1234
55811730
```

Live Test

- 30 minutes warm-up
- 3 hours Q1 - Q6
- 30 minutes mix-Q1Q2Q3Q4Q5Q6
- Preparing for the live test
  - Choose a database based on your observations from previous phases and all six queries
  - Caching known requests will not work(unless you are smart)
  - Need to have all Qs running at the same time
  - Don’t expect testing in sequence
  - Avoid bottlenecks in mixed queries

Warnings

- Avoid tagging penalties
- Keep a watch on budget.
  - $55 (phase + livetest)
- Check correctness before livetest
- Start early