15-319 / 15-619 Cloud Computing

Recitation 11

April 4th 2017

Overview

- **Last week's reflection**
  - Team Project, Phase 1, Queries 1, 2 & 3
  - Unit 4 - Modules 16 and 17
  - Quiz 9
- **This week's schedule**
  - Team Project, Phase 1, report
  - Team Project, Phase 2, Queries, 1, 2, 3 & 4
  - Project 4.1, Batch Processing with MapReduce
  - Unit 5- Module 18
  - Quiz 10
- **Twitter Analytics: The Team Project**

Reminders

- Monitor AWS expenses regularly and tag all resources
  - Check your bill both on AWS and TPZ
- Piazza Guidelines
  - Please tag your questions appropriately
  - Search for an existing answer first
- Provide clean, modular and well documented code
  - Large penalties for not doing so.
  - **Double check** that your code is submitted!!
    (verify by downloading it from TPZ from the submissions page)
- Utilize Office Hours
  - We are here to help (but not to give solutions)

Conceptual Module to Read

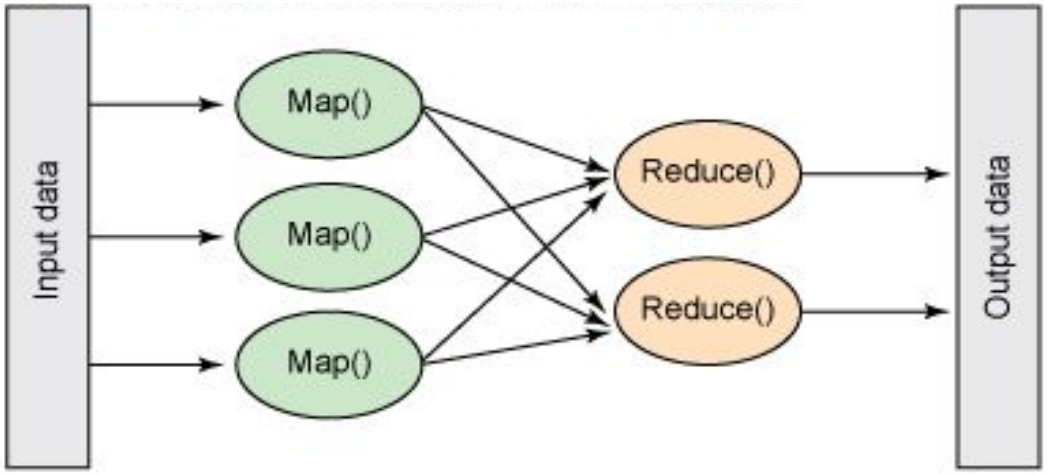
- UNIT 5: Distributed Programming and Analytics Engines for the Cloud
  - Module 18: Introduction to Distributed Programming for the Cloud
  - Module 19: Distributed Analytics Engines for the Cloud: MapReduce
  - Module 20: Distributed Analytics Engines for the Cloud: Spark
  - Module 21: Distributed Analytics Engines for the Cloud: GraphLab
  - Module 22: Message Queues and Stream Processing

Project 4

- Project 4.1, Batch Processing with MapReduce
  - MapReduce Programming
- Project 4.2
  - Iterative Programming Using Apache Spark
- Project 4.3
  - Stream Processing using Kafka/Samza

Introduction to MapReduce

- **Definition**: Programming model for processing large datasets with a parallel, distributed algorithm on a cluster
- **Phases of MapReduce:**
  - **Map**
  - **Shuffle**
  - **Reduce**

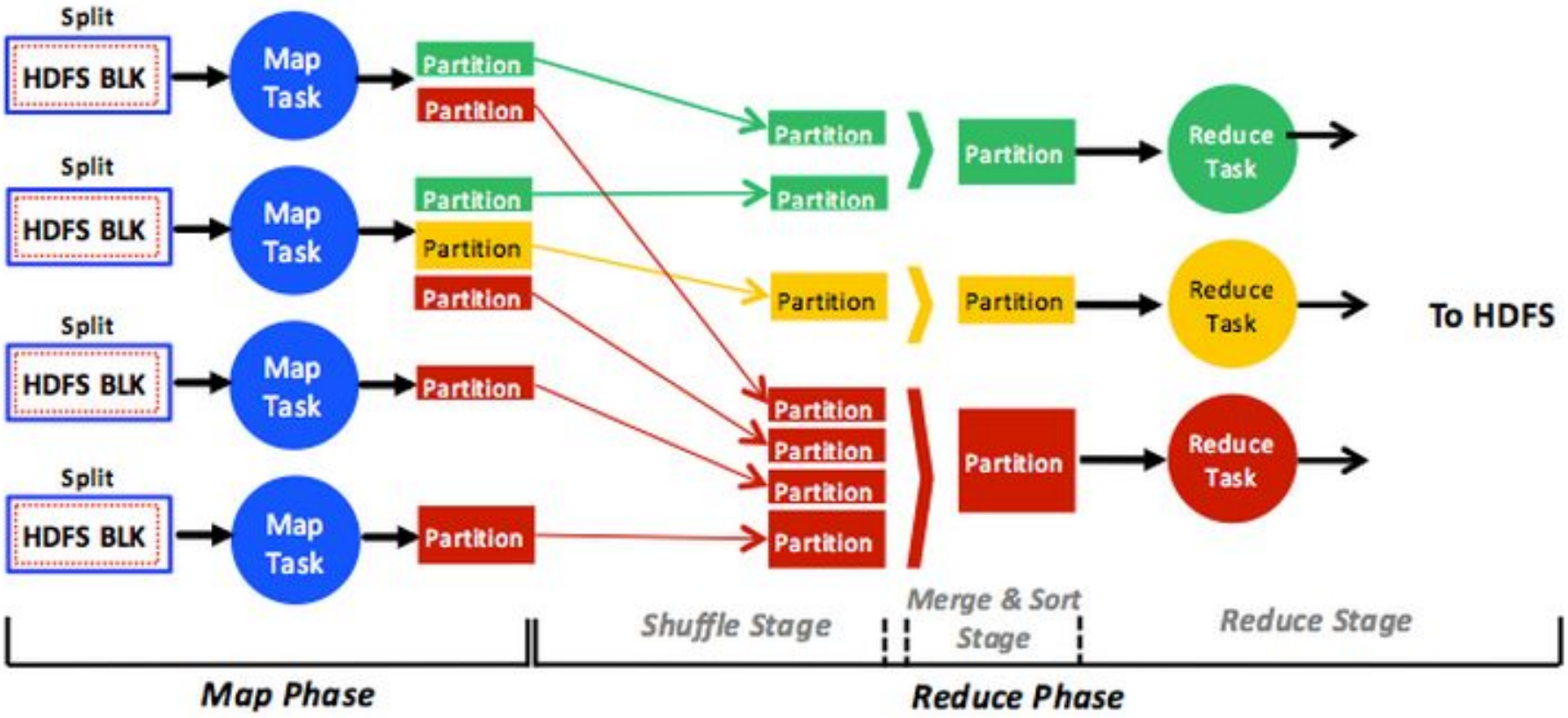


MapReduce: Framework

- The MapReduce framework:
  - Partitions the input data
  - Schedules the program's execution across a set of machines
  - Performs the group by key (sort & shuffle) step
  - Handles machine failures
  - Manages required inter-machine communication

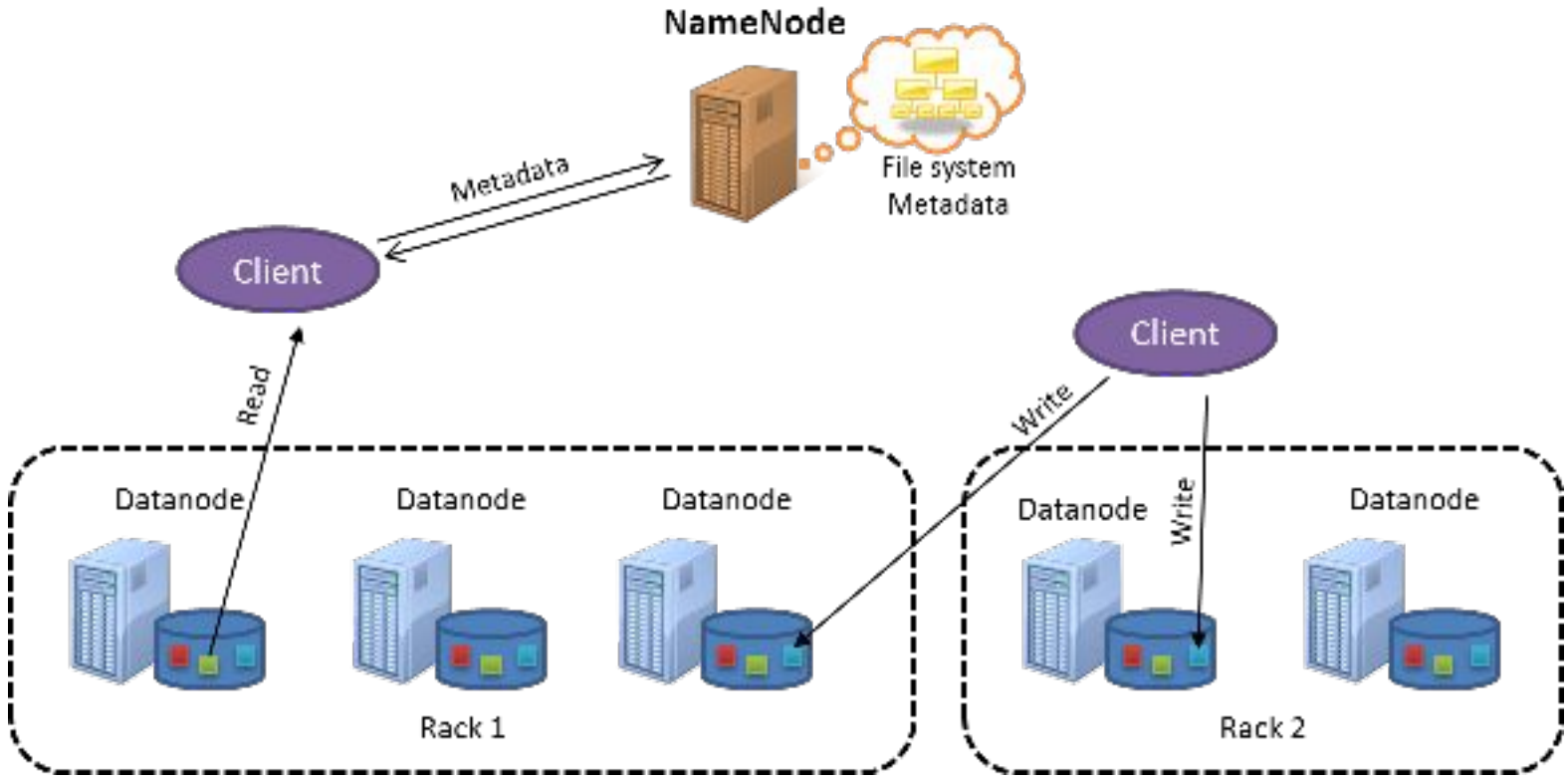
MapReduce and HDFS

- Detailed workflow



HDFS - Distributed File System

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



MapReduce Data Types - 1

- 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)

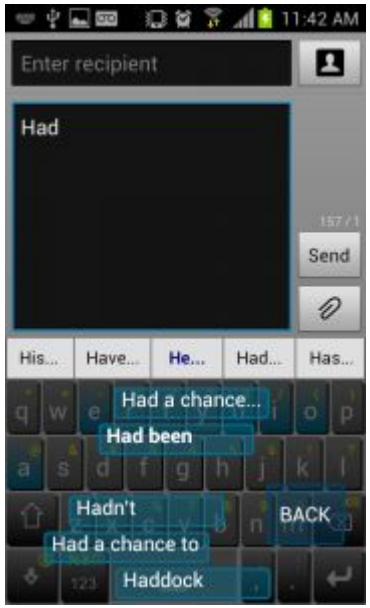
MapReduce Data Types - 2

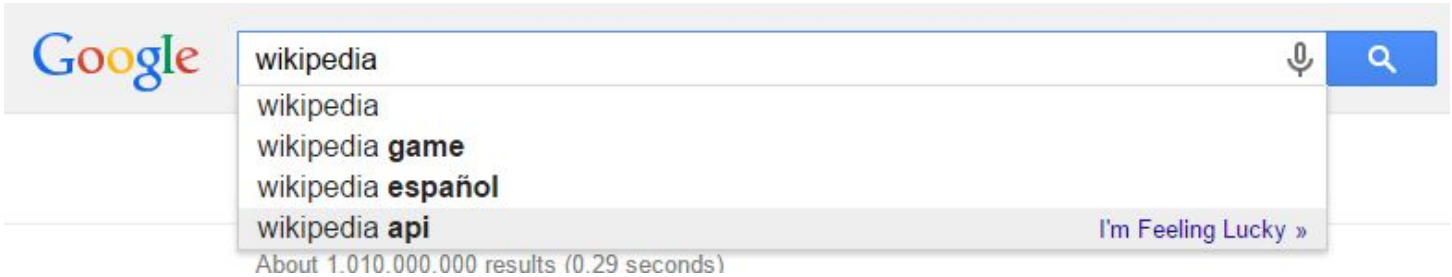
- Reducer
  - Input: **key-value pairs**
    - A list of values for each key output from the mapper
  - Output: **key-value pairs**
    - The desired result from your aggregation

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

Project 4.1 - Input Text Predictor

- Suggest words based on phrases already typed
  - Use an English Corpus to build a language model





Project 4.1

- Steps to build an Input Text Predictor
  - Utilize raw XML dataset of English text
  - Extract XML content and clean the input data
  - Perform the N-Gram count
  - Build the Statistical Language Model
  - Predict the next word given a phrase
- Have to use a Custom JAR in EMR
  - **CANNOT use EMR Streaming as in P1.2**

Construct an Input Text Predictor - 1

1. Given a language corpus
   - Wikipedia dataset (~7.8 GB)
   - Clean the dataset
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 the probability data to HBase and Redis, and predict the next word based on the probabilities

P4.1 Bonus

- MapReduce for phrase auto-completion
  - Given prefix, suggest the most possible phrases
  - Example: given “carnegie”,
    - Possible phrases are: carnegie mellon, carnegie library,carnegie mellon university faculty….
    - Suggest at most 8 phrases with highest probability, and make suggestions on all possible lengths using the n-gram data.
    - Three 1-gram, two 2-grams, two 3-grams and one 4-gram
  - Store probability data to HBase and connect our front-end to submit
- Worth
  - 10% autograded
  - 5% for two follow up tasks, manually graded

Recommendations

- Test for correctness with a small dataset first
- Task 1 submission could take a long time, plan ahead and know what you are supposed to do before submission
- **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
  - The result of task 1 is useful for task 2 and bonus!
- Version of Hadoop
  - It should match the version shown in the EMR AMI
- Start early and try the bonus

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

Project 4.1 FAQ

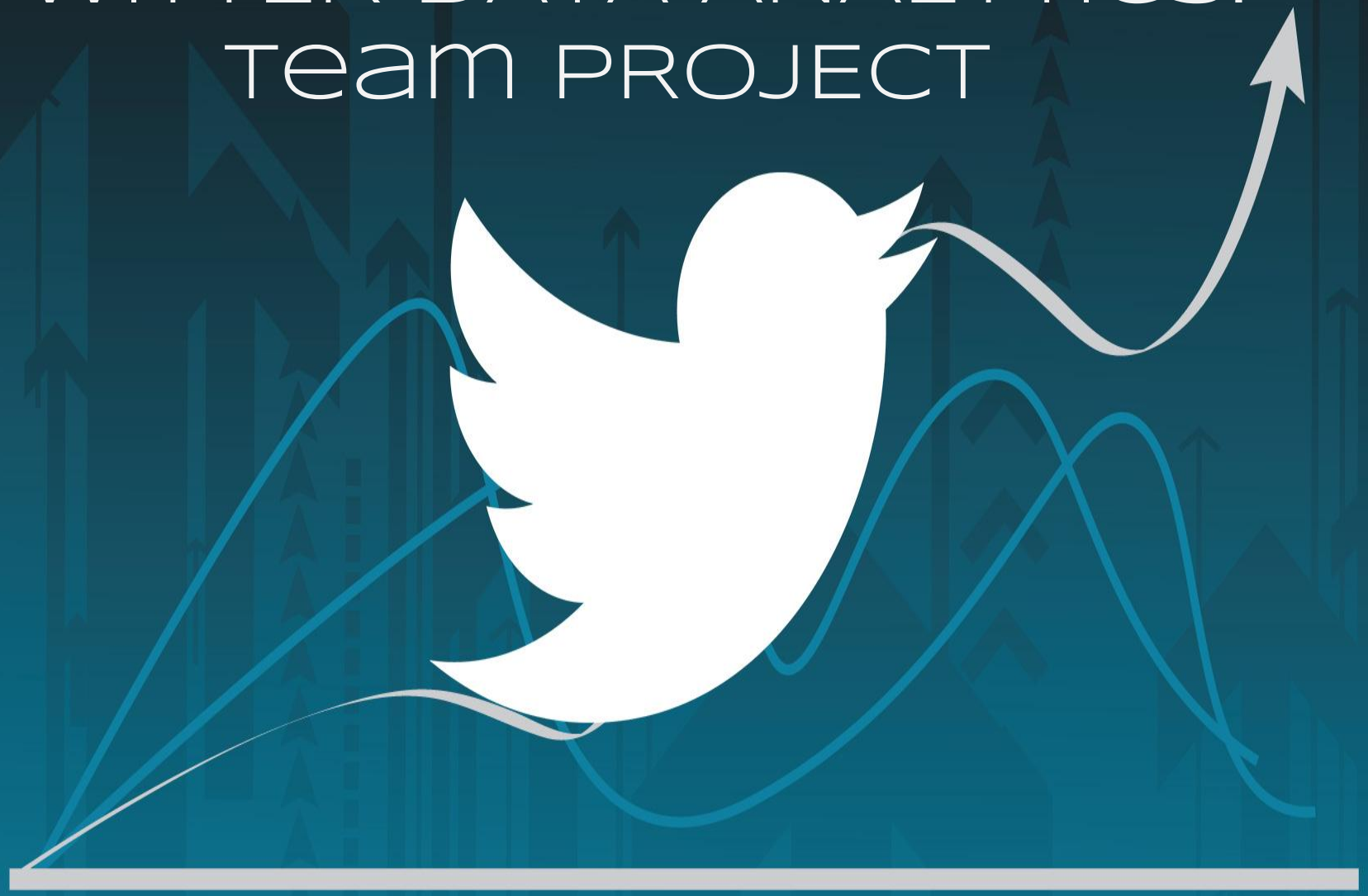
- MapReduce job failures or errors
  - Always test locally first
  - On step failure, continue or terminate?
  - Hadoop UI
    - Find job logs
    - Locate the source of the problem
    - Solve the problem
  - For memory problems:
    - Restrict the number of mappers and reducers
    - Increase the Java program memory

Project 4.1 FAQ

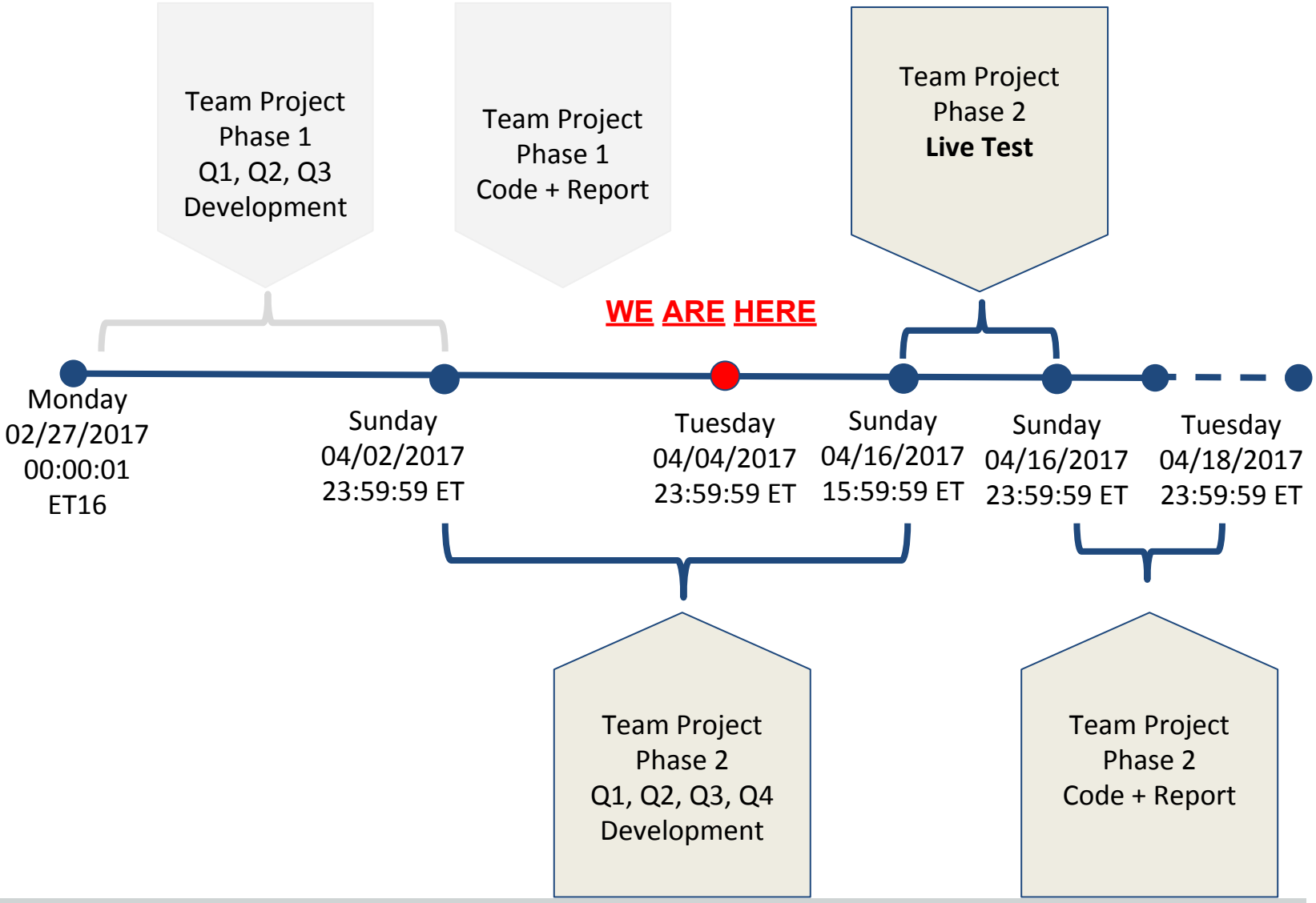
- Could not get a full score on the tasks
  - Test your regex…for some corner cases!
  - Incorrect use of combiner
    - Filtered words in Combiner and Reducer

Questions?

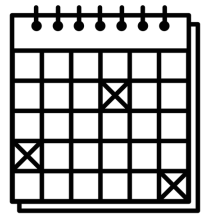
TWITTER DATA ANALYTICS: Team PROJECT



Team Project Phase 2 Deadlines



Team Project Time Table



Phase	Query	Start	Deadline	Code and Report Due
Phase 1	Q1	Monday 2/27/2017 00:00:01 EST	Sunday 3/12/2017 23:59:59 EST	-
	Q2 & Q3	Monday 2/27/2017 00:00:01 EST	Sunday 4/2/2017 23:59:59 EST	Tuesday 4/4/2017 23:59:59 EST
Phase 2	**Q1, Q2, Q3, Q4**	**Monday 4/3/2017 00:00:01 EST**	**Sunday 4/16/2017 15:59:59 EST**	**-**
	Live Test Q1, Q2, Q3, Q4	**Sunday 4/16/2017 18:00:01 ET**	**Sunday 4/16/2017 23:59:59 EST**	**Tuesday 4/18/2017 23:59:59 EST**
Phase 3	**Q1, Q2, Q3, Q4, Q5**	**Monday 4/17/2017 00:00:01 ET**	**Sunday 4/30/2017 15:59:59 EST**	**-**
	Live Test Q1, Q2, Q3, Q4, Q5	**Sunday 4/30/2017 18:00:01 ET**	**Sunday 4/30/2017 23:59:59 EST**	**Tuesday 5/2/2017 23:59:59 EST**

Phase 1 Review

- 3 Queries
  - 1 computing-focused query
  - 2 database-backed query
- Get familiar with
  - Doing efficient ETL
  - Designing good schema for both backends
  - Building performant and robust frontend
- Next step is to further tune your backend and frontend
  - Find out the expensive parts of handling a query
  - Think about possible ways to speed up
  - Try it out!!!
- Code and report are due tonight (April 4)!!

Phase 2

- Phase 2 accounts for 30% of the total score of the Team Project
  - Phase 1 only accounts for 20%
- You need to continue exploring MySQL and HBase in phase 2
  - You will continue to work on Q1, Q2 & Q3
  - A new query, Q4, is added
- **You must achieve over 80% correctness**, **<u>and</u> at least 50% RPS in both MySQL and HBase** in order to get points for each query.
- Your performance RPS is **<u>SOLELY</u>** determined by the Live-Test
  - Some students cached query results at front-end in phase 1
    - If not done wisely, this may lead to the front-end crashing during the Phase 2 Live-Test
- As before, a report needs to be submitted for Phase 2 after the Live-Test
  - Check the schedule for deadlines

Phase 2

- Budget on AWS for Phase 2 is $50 (including the Live Test), if you spend more than $75 you will receive a 100% penalty.
- You have a $0.88/hr budget for MySQL and HBase separately in the Live Test. You will receive (X-0.88)*2% penalty if you spend X dollars > $0.88. The hourly cost includes:
  - EC2
    - We evaluate your cost using the On-Demand Pricing towards **$0.88/hour** even if you use spot instances.
  - EBS & ELB
  - Ignore data transfer and EMR cost
- We encourage you to do ETL on Azure and GCP to save your budget on AWS

Phase 2 Live Test - Hbase

Time	Value	Target	Weight
04/16 at 3:59pm	Deadline to submit the DNS of your Web Service		
6:00pm - 6.25pm	Warm-up (Q1 only)	0	0%
6:25pm - 6:50pm	Q1	30000	4%
6:50pm - 7:15pm	Q2	11000	8%
7:15pm - 7:40pm	Q3	2500	8%
7:40pm - 8:05pm	Q4	TBD	8%
8:05pm - 8:30pm	Mixed (Q1,Q2,Q3,Q4)	7500/2800/700/TBD	3+3+3+3=12%

- You need to achieve at least 50% of the target RPS and 80% correctness for BOTH MySQL and HBase to get a score for a query!
- Report is worth 20% of the grade in this phase!

Phase 2 Live Test - MySQL

Time	Value	Target	Weight
9:00pm - 9.25pm	Warm-up (Q1 only)	0	0%
9:25pm - 9:50pm	Q1	30000	4%
9:50pm - 10:15pm	Q2	11000	8%
10:15pm - 10:40pm	Q3	2500	8%
10:40pm - 11:05pm	Q4	TBD	8%
11:05pm - 11:30pm	Mixed (Q1,Q2,Q3,Q4)	7500/2800/700/TBD	3+3+3+3=12%

- You need to achieve at least 50% of the target RPS and 80% correctness for BOTH MySQL and HBase to get a score for a query!
- Report is worth 20% of the grade in this phase!

Query 4: Interactive Tweet Server

There are 7 different parameters in the request URL for a request

/q4?op=<operation>&field=<field name>&tid1=<tweet id1>&tid2=<tweet id2>&payload=<value>&uuid=<unique id>&seq=<sequence number>

General Info:

1. Four operations:
   - write, set, read and delete
2. Operations under the same **uuid** should be executed in the order of the sequence number.
3. Be wary of malformed queries.

Query 4: Interactive Tweet Server

field	type	example
tweetid	long int	15213
userid	long int	156190000001
username	string	CloudComputing
timestamp	string	Mon Feb 15 19:19:57 2017
text	string	Welcome to P4!#CC15619#P3
favorite_count	int	22
retweet_count	int	33

Query 4: Interactive Tweet Server

- Write Request

  `/q4?op=write&field=<empty>&tid1=<empty>&tid2=<empty>&payload=json_string&uuid=unique_id&seq=sequence_number`

- Response

  `TEAMID,TEAM_AWS_ACCOUNT_ID\n`

  `success\n`

- `payload` is the url-encoded json string; same structure as the original tweet json; only contains the seven fields needed. For tid and uid, don't get them from the "id_str" field, only get them from the "id" field.

Query 4: Interactive Tweet Server

- Read Request

  `/q4?op=read&field=<empty>&tid1=tweet_id1&tid2=tweet_id2&payload=<empty>&uuid=unique_id&seq=sequence_number`

- Response

  `TEAMID,TEAM_AWS_ACCOUNT_ID\n tid_n\ttimestamp_n\tuid_n\tusername_n\ttext_n\tfavorite_count_n\tretweet_count_n\n`

- Range read

Query 4: Tweet Server

- Delete Request

  `/q4?op=delete&field=<empty>&tid1=tweet_id&tid2=<empty>&payload=<empty>&uuid=unique_id&seq=sequence_number`

- Response

  `TEAMID,TEAM_AWS_ACCOUNT_ID\n`

  `success\n`

- Delete the whole tweet

Query 4: Tweet Server

- Set Request

  `/q4?op=set&field=field_to_set&tid1=tweet_id&tid2=<empty>&payload=string&uuid=unique_id&seq=sequence_number`

- Response

  `TEAMID,TEAM_AWS_ACCOUNT_ID\n`

  `success\n`

- Set one of the text, favorite_count, retweet_count of a particular tweet
- Payload is url-encoded

Query 4: Tweet Server

- Malformed Request

  `/q4?op=set&field=field_to_set&tid1=tweet_id&tid2=<empty>&payload=0;drop+tables+littlebobby&uuid=unique_id&seq=sequence_number`

- Response

  `TEAMID,TEAM_AWS_ACCOUNT_ID\n`

  `success\n`

Query 4: Tweet Server

- Part of the queries
  - To make debugging easier, there will always be a Write or Delete of a tweet before a Read
  - But some of the tweets are read only, we will only submit read requests on these tweets
  - For the Live Test, we will not follow the above rule so don’t rely on the above to get a high RPS

Team Project General Hints

- Identify the bottlenecks using fine-grained profiling.
- Do not cache naively.
- Review what we have learned in previous project modules
  - Scale out
  - Load balancing (are requests balanced?)
  - Replication and sharding
- Look at the feedback of your Phase 1 report!

Team Project, Q4 Hints

- Start with one machine if you are not sure that your concurrency model is correct. Pay attention to scalability.
- Adopt a forwarding mechanism or a non-forwarding mechanism
  - You may need a custom load balancer
- May need many connections at the same time, in the case of out of order sequence numbers.
- Consider batch writes. Write and read are exclusive due to the consistency model.

Upcoming Deadlines

- Team Project: Phase 1, Q1, Q2M, Q2H, Q3M, Q3H
  - Phase 1 **code and report** due on 04/04/2017 
- Quiz 10: Unit 5 - Module 18
  - Due: 04/07/2017 11:59 PM Pittsburgh 
- Project 4.1: Batch Processing with MapReduce
  - Due: 04/09/2017 11:59 PM Pittsburgh 
- Team Project: Phase 2, Q1, Q2, Q3, Q4 (M&H)
  - Live-test DNS due: 04/16/2017 3:59 PM Pittsburgh
  - Code and report due:04/18/2017 11:59 PM Pittsburgh

Questions?