15-319 / 15-619 Cloud Computing

Recitation 14

April 24th 2018

Overview

- **Last week's reflection**
  - Project 4.2
- **This week's schedule**
  - Team Project, Phase 3
  - Project 4.3
    - Released on Saturday 4/28
    - Due on Friday 5/4
- **Twitter Analytics: The Team Project**
  - Phase 3
    - Live Test on Sunday 4/29

Project 4

- Project 4.1
  - Batch Processing with MapReduce
- Project 4.2
  - Iterative Batch Processing Using Apache Spark
- Project 4.3
  - Stream Processing with Kafka and Samza

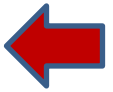
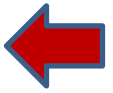
P4.2 Reflection

- Programming in Scala on Spark
- Understanding the differences between processing data with MapReduce and Spark
- Exploring Twitter social data with RDD and SparkSQL APIs
- Implementing an iterative processing algorithm - pagerank - on a large dataset
- Utilizing the Spark Web UI to monitor a Spark job and identify performance bottlenecks
- Tuning a Spark program to optimize for time

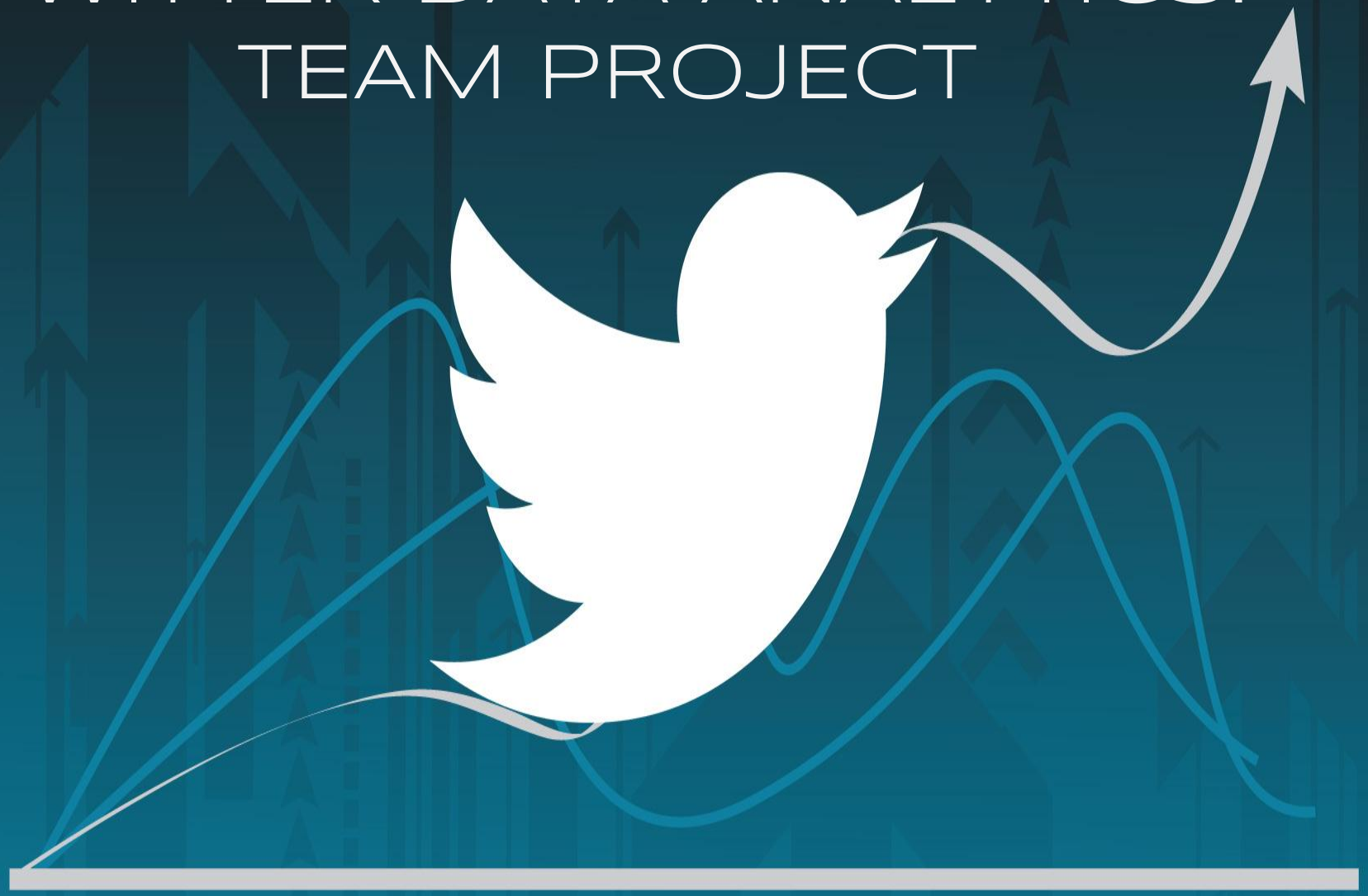
P4.2 Reflection

- Common Issues
  - Handling dangling nodes in the graph
  - Out of memory errors
  - Long running jobs
    - Reduce the amount of data shuffling
- Takeaways
  - Some approaches to implementing pagerank are more efficient than others
  - The Spark Web UI is a useful visualization tool

Upcoming Deadlines

- Team Project : Phase 3 
  - Live Test: 4/29/2018 3:59 PM Pittsburgh
  - Code and report due: 05/1/2018 11:59 PM Pittsburgh
- Project 4.3 : Stream Processing with Kafka/Samza 
  - Released: **Saturday, 04/28/2018**
  - Due: 05/04/2018 11:59 PM Pittsburgh

TWITTER DATA ANALYTICS: TEAM PROJECT



Query 4: Interactive Tweet Server

You are going to build a web service that supports READ, WRITE, SET and DELETE requests on tweets.

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
4. Only English tweets are needed

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&payload=json_string&uuid=unique_id&seq=sequence_number`

- Response

  `TEAMID,TEAM_AWS_ACCOUNT_ID\n`

  `success\n`

- `payload`
  - It is a url-encoded json string;
  - It has the same structure as the original tweet json;
  - It 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&uid1=userid_1&uid2=userid_2&n=max_number_of_tweets&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`

Query 4: Tweet Server

- Delete Request

  `/q4?op=delete&tid=tweet_id&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&tid=tweet_id&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 a url-encoded string

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`

Phase 2 Live Test Issues

- Cache became too large
- Used up IOPS
- Forgot to catch exceptions

Team Project General Hints

- Identify the bottlenecks using fine-grained profiling.
- Do not cache naively.
- Use logging to debug concurrent issues
- Review what we have learned in previous project modules
  - Load balancing (are requests balanced?)
  - Replication and sharding
  - Multi-threading programming
- Look at the feedback of your Phase 1 and Phase 2 reports!
- To test mixed queries, run your own load generator.
  - Use jmeter/ab/etc.
- For the successful teams, they usually have a good testing strategy.

Team Project, Q4 Hints

- Start with one machine if you are not sure that your concurrency model is correct.
- Adopt a forwarding mechanism or a non-forwarding mechanism
  - You may need a custom load balancer
- Think carefully about your async/sync design
- May need many connections and threads at the same time, in the case of out of order sequence numbers.
- Spare enough time for Q4 as you will face deadlock issues that are hard to resolve.

Team Project Time Table

Phase (and query due)	Start	Deadlines	Code and Report Due
Phase 1 • Q1, Q2	Monday 02/26/2018 00:00:00 ET	**Checkpoint 1, Report: Sunday 03/11/2018 23:59:59 ET** **Checkpoint 2, Q1: Sunday 03/25/2018 23:59:59 ET** **Phase 1, Q2: Sunday 04/01/2018 23:59:59 ET**	Phase 1: Tuesday 04/03/2018 23:59:59 ET
Phase 2 • Q1, Q2,Q3	Monday 04/02/2018 00:00:00 ET	**Sunday 04/15/2018 15:59:59 ET**	
Phase 2 Live Test (Hbase **AND** MySQL) • Q1, Q2, Q3	Sunday 04/15/2018 17:00:00 ET	**Sunday 04/15/2018 23:59:59 ET**	**Tuesday 04/17/2018 23:59:59 ET**
Phase 3 • Q1, Q2, Q3, Q4	Monday 04/16/2018 00:00:00 ET	**Sunday 04/29/2018 15:59:59 ET**	
Phase 3 Live Test (Hbase **OR** MySQL) • Q1, Q2, Q3, Q4	**Sunday 04/29/2018 17:00:00 ET**	**Sunday 04/29/2018 23:59:59 ET**	**Tuesday 05/01/2018 23:59:59 ET**

Questions?