15-319 / 15-619
Cloud Computing

Recitation 9

October 27, 2020

Overview

- **Last week's reflection**
  - Project 3.3
  - Query 2 Checkpoint, Team Project
  - OLI Unit 4 Module 14: Cloud Storage
  - Quiz 7
- **This week's schedule**
  - OLI - Modules 15,16,17
    - Quiz 8 – due on Friday, Oct 30
- **Team Project, Phase 1**
  - Query 2 is due on Sunday, Nov 1
  - Final report is due on Tuesday, Nov 3

Project 3.3 Reflection

- You have explored
    - Sharding and replication
    - Multithreaded programming
    - Strong consistency model
        - Use PRECOMMIT to keep proper order on all datastores
    - Bonus Task: Eventual Consistency
        - No guarantee of ordering for incoming requests
        - Compare timestamp with last timestamp for the key

Project 3.3 Reflection

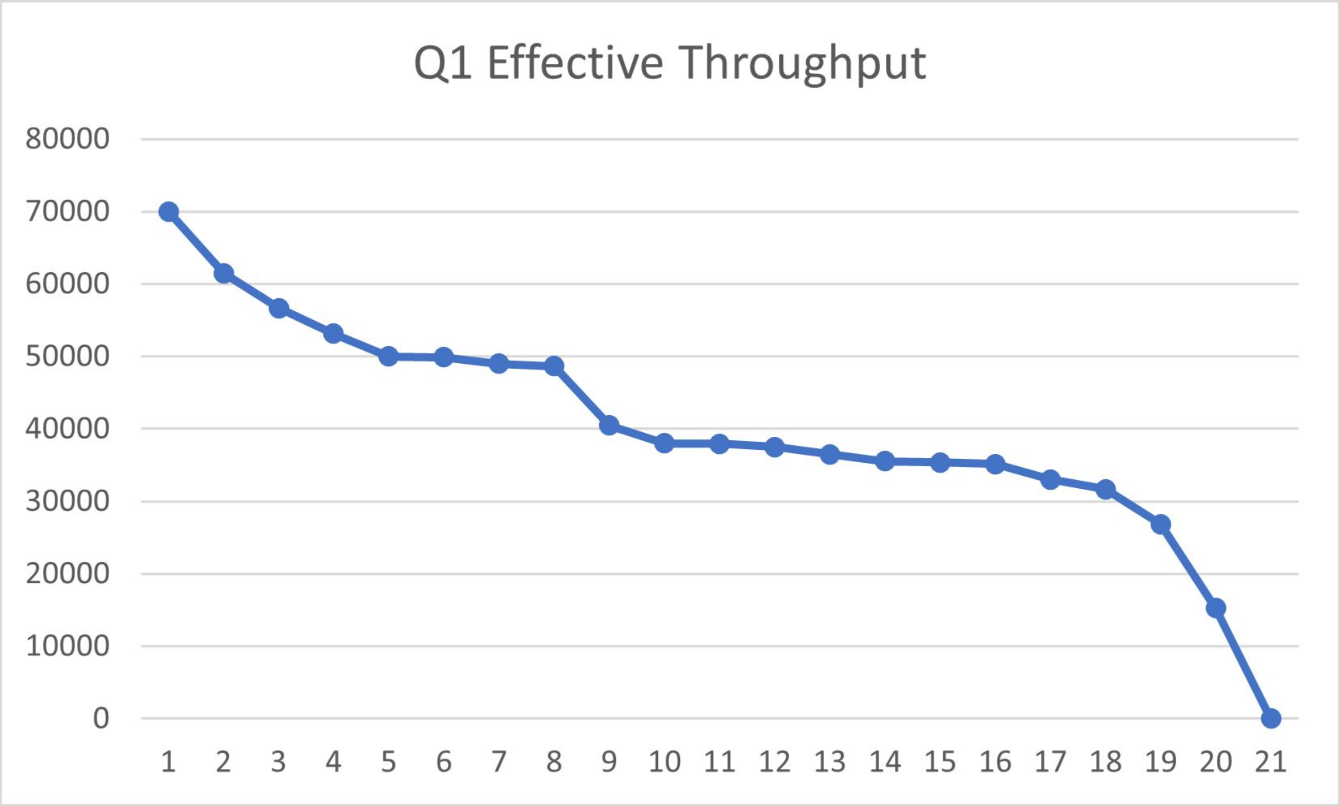
- Most common issues:
    - Incorrect implementation of locking using a Priority Queue
    - Incorrect use of wait() and notifyAll()
    - Improper implementation of synchronization block
    - Exception in threads, which caused the threads to exit prematurely without closing the connection

- Best ways to debug:
    - Logging to keep track of what exactly happened with a request

TEAM PROJECT
Twitter Data Analytics

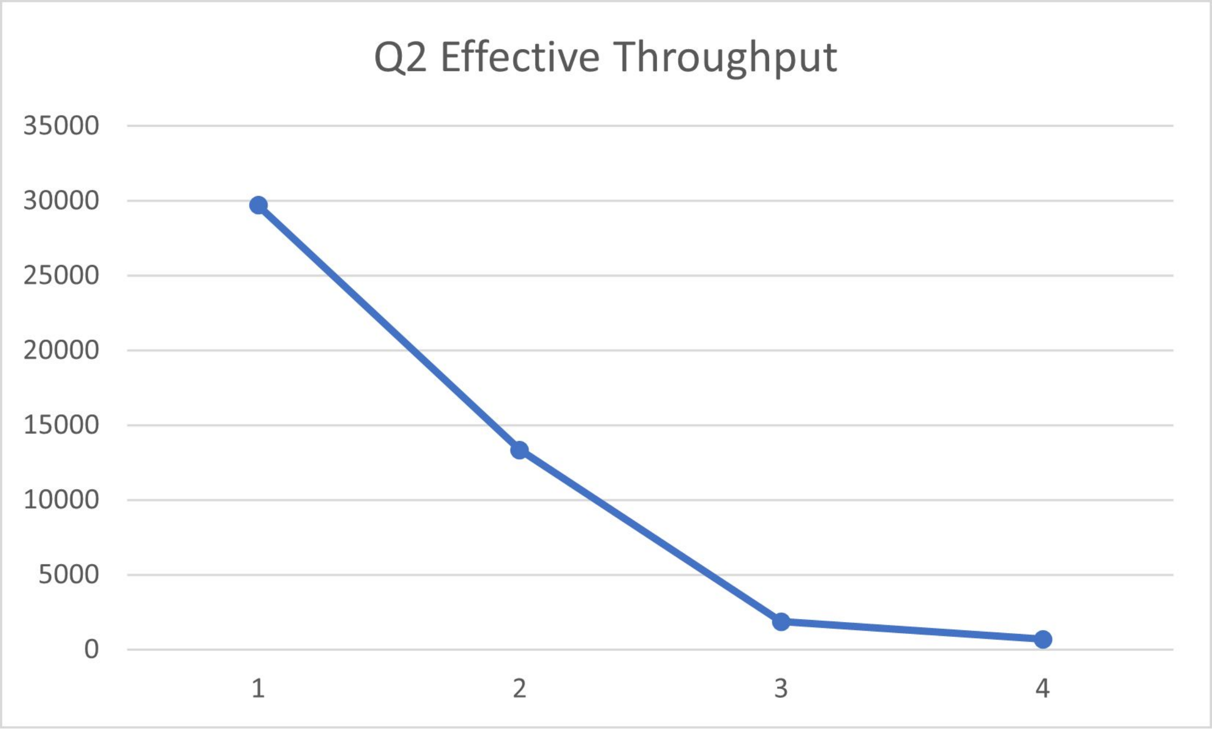
Team Project - Query 1

- 17/21 teams reached 32,000 RPS.



Team Project - Query 2

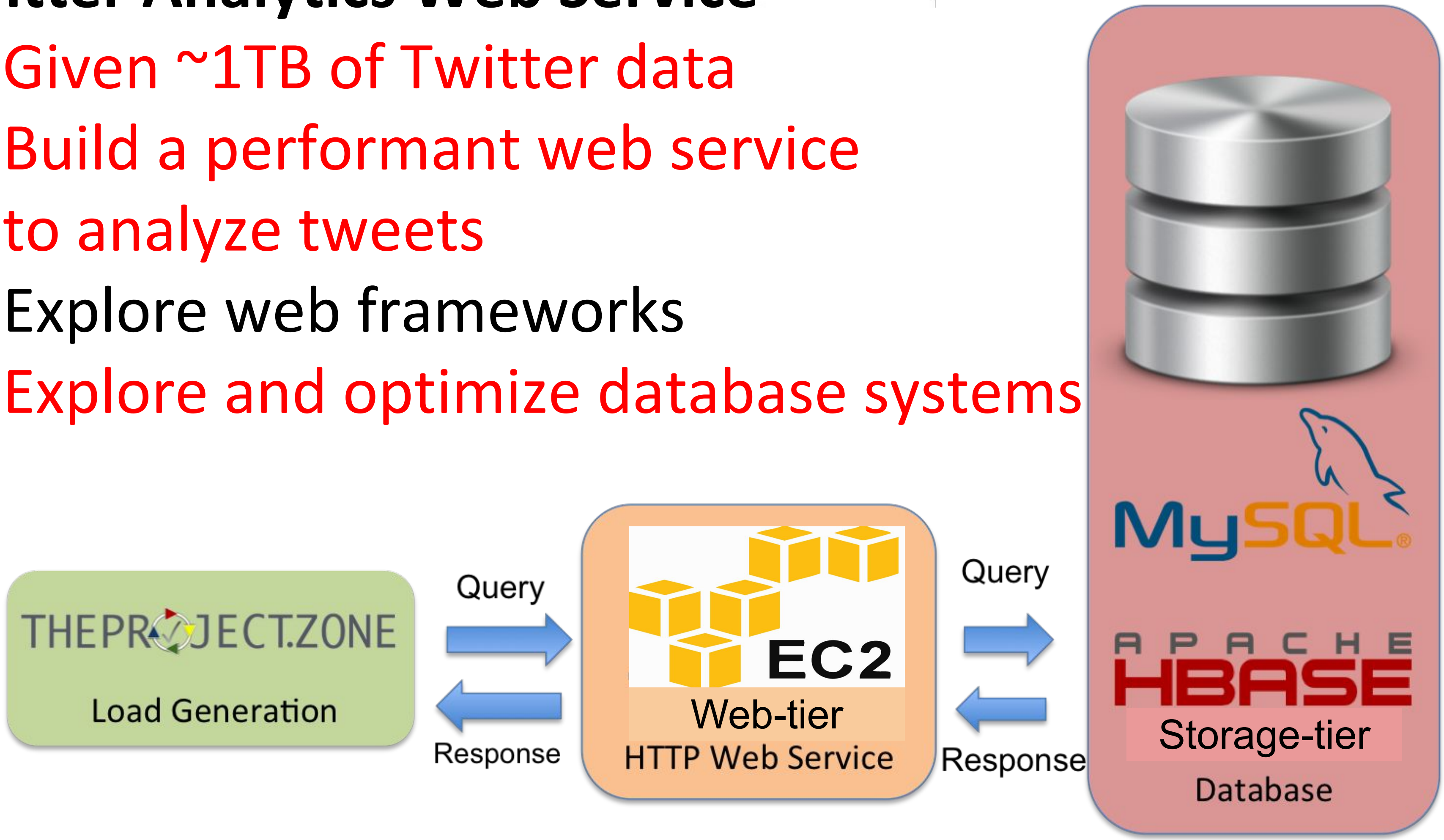
- 2 teams have non-zero score for MySQL
- 1 team has non-zero score for HBase
- Team Jedi reached 10,000 RPS for both MySQL&HBase



Team Project - Query 2

Twitter Analytics Web Service

- Given ~1TB of Twitter data
- Build a performant web service to analyze tweets
- Explore web frameworks
- Explore and optimize database systems



Load Generation

Query

Response

Web-tier

HTTP Web Service

Query

Response

Storage-tier

Database

Query 2 - User Recommendation System

target throughput: 10,000 RPS **for both MySQL and HBase**

Use Case: When you follow someone on twitter, recommend users you may also be interested in

Query: GET /q2?**user_id**=<ID>&**type**=<TYPE>&**phrase**=<PHRASE>&**hashtag**=<HASHTAG>

Response:

```
<TEAMNAME>,<AWSID>\n
uid\tname\tdescription\ttweet\n
uid\tname\tdescription\ttweet
```

Three Scores:

- Interaction Score - closeness
- Hashtag Score - common interests
- Keywords Score - match specific interests

Final Score: Interaction Score * Hashtag Score * Keywords Score

Load Data & Backup

Hint: One way to load data into MySQL and HBase databases is to load from a tsv file. Please refer *Load Data* part in MySQL Primer, *HBase Java API* part of HBase Primer and *Load Data* part in Project 3.1.

Once this step is completed, you should backup your database to save cost. There are various ways for you to backup your MySQL database, e.g. mysqldump. For HBase, you can backup HBase database on S3 using the hbase snapshot.

Be very careful about escape characters.

- For example, how you will treat \n (new line) and real backslash \

Be very careful about encodings.

- The tweets contain a lot of languages and even emojis

Performance Tuning Tips

- To do performance tuning, you first need to identify which part of your system is the bottleneck
    - Profile and monitor your system
        - Use CloudWatch for resource utilization such as CPU, Network, Disk, etc.

Performance Tuning Tips

- Web Tier
  - Blocking v.s Non-Blocking?
  - Is the workload distributed evenly on multiple web servers?
  - Have you optimized your code?

    StringBuilder vs +
  - Did you put too much computation in the web tier when this can be precomputed in ETL?

Performance Tuning Tips

- Database Tier - MySQL
    - Different MySQL engines

- Database Tier - HBase
    - Locality and compaction, region server split, etc
    - Scan can be really slow, try to avoid it if possible
      If you can’t, try to scan as few rows as possible
- Tune parameters
    - Check the official documentation
    - Search for performance tuning best practices

If your throughput is only 50% of the target throughput, don’t invest your time in parameter tuning, it’s not magic.
A good schema can easily double or even triple the target throughput using the default parameters without any parameter tuning. Please focus on schema design first!

Reminders on penalties

- M family instances **only**, smaller than or equal to **large** type
- Other types are allowed (e.g., t2.micro) **but only for testing**
  - Using these for any submissions = 100% penalty
- Only General Purpose (gp2) SSDs are allowed for storage
  - e.g **m5d is not allowed** since it uses NVMe storage
- AWS endpoints only (EC2/ELB).
- **$0.70/hour (MySQL) and $0.85/hour (HBase)** applies to every submission

Phase 1 Budget

- Your web service should not cost more than **$0.70/hour (Q1 and Q2 MySQL) and $0.85/hour (Q2 HBase)** this includes:
  - EC2 cost (Even if you use spot instances, we will calculate your cost using the **on-demand** instance price)
  - **EBS cost**
  - **ELB cost**
  - We will not consider the cost of data transfer and EMR
  - See writeup for details
- AWS total budget of $55 for Phase 1

Suggested Tasks for Phase 1

Phase 1 weeks	Tasks	Deadline
Week 1 ● 10/12 - 10/18	● Team meeting ● Read Writeup & Report ● Complete Q1 code & achieve correctness ● Start ETL on mini dataset and design q2 schema	● Q1 Checkpoint due on 10/18 ● Checkpoint Report due on 10/18
Week 2 ● 10/19 - 10/25	● Q1 target reached ● Q2 ETL & Initial schema design completed ● Achieve Q2 basic correctness and submit to TPZ	● Q1 final target due on 10/25 ● Q2 MySQL Checkpoint due on 10/25 ● Q2 HBase Checkpoint due on 10/25
Week 3 ● **10/26-11/1**	● **Achieved correctness for both Q2 MySQL, Q2 HBase & basic throughput** ● **Optimizations to achieve target throughputs for Q2 MySQL and Q2 HBase**	● Q2 MySQL final target due on 11/1 ● Q2 HBase final target due on 11/1 ● Final Report due on 11/3

Piazza Team Project Hint Thread

We will keep posting hints and clarifications in this thread, please check it frequently

https://piazza.com/class/kckujccg5497i0?cid=922