15-319 / 15-619
Cloud Computing

Recitation 7

October 13, 2020

Overview

- **Last week's reflection**
  - Project 3.1
  - OLI Unit 3 - Module 10, 11, 12
  - Quiz 5
- **This week's schedule**
  - Project 3.2
  - OLI Unit 3 - Module 13
  - Quiz 6 (Due on Friday, October 16)

Last Week

- **Unit 3: Virtualizing Resources for the Cloud**
  - Module 10: Resource Virtualization - Memory
  - Module 11: Resource Virtualization - I/O
  - Module 12: Case Study
- **Quiz 5**
- **Project 3.1**
  - Files v/s Databases (SQL & NoSQL)
    - Flat files
    - MySQL
    - Redis & Memcached
    - HBase

This Week

- **OLI : Unit 3 Module 13 -** Storage and Network Virtualization
- **Quiz 6** - October 16, 2020
- **Project 3.2** - Sunday, October 18
  - Social Networking Timeline with Heterogeneous Backends
    - MySQL
    - Neo4j
    - MongoDB
    - Caching
    - Choosing Databases, Storage Types & Tail Latency
- **Online Programming Exercise for Multi-Threading on Cloud9**
- **Team Project, Phase 1 released -** Monday, October 12

This Week's Conceptual Content

- **Unit 3 - Module 13: Storage and network virtualization**
  - Software Defined Data Center (SDDC)
  - Software Defined Networking (SDN)
    - Device virtualization
    - Link virtualization
  - Software Defined Storage (SDS)
    - IOFlow
- **Quiz 6**

Open Learning Initiative

Transforming higher education through the science of learning.

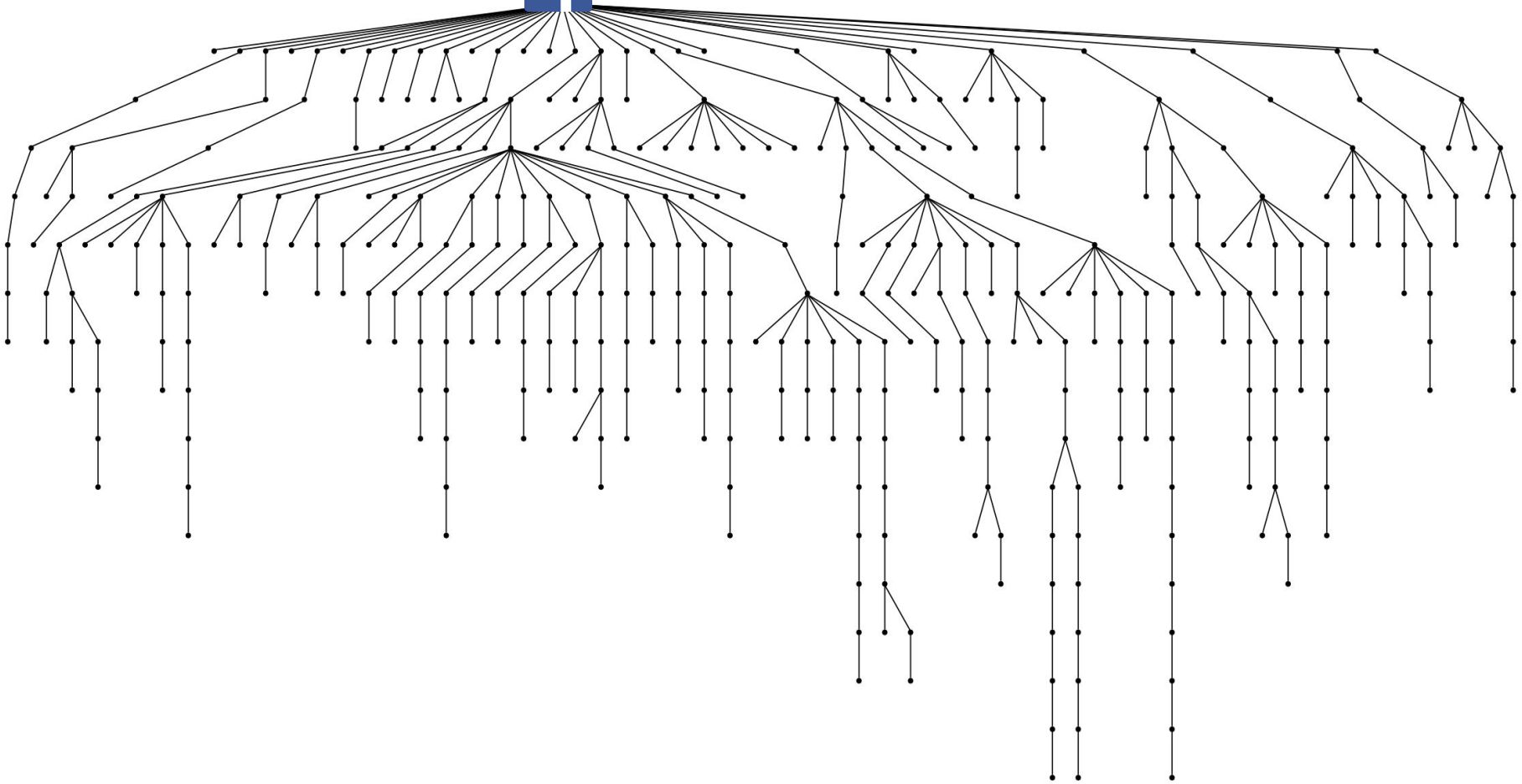
Individual Projects

- DONE
  - P3.1: Files vs Databases - comparison and Usage of flat files, MySQL, Redis, and HBase
  - NoSQL Primer
  - HBase Basics Primer
  - MongoDB Primer
- **NOW**
  - P3.2: Social networking with heterogeneous backends
- Coming Up
  - P3.3: Multi-threading Programming and Consistency

A Social Network Service

High Fanout in Data Fetching

A single page, requires many data fetch operations



Nishtala, R., Fugal, H., Grimm, S., Kwiatkowski, M., Lee, H., Li, H. C., ... & Venkataramani, V. (2013, April). Scaling Memcache at Facebook. In *nsdi* (Vol. 13, pp. 385-398).

Neo4j

- Designed to treat the relationships between data as equally important as the data
  - Relationships are very important in social graphs
- Property graph model
  - Nodes
  - Relationships
  - Properties
- Cypher query language
  - Declarative, SQL-inspired language for describing patterns in graphs visually

MongoDB

- Document Database
  - Schema-less model
- Highly Scalable
  - Automatically shards data among multiple servers
  - Does load-balancing
- Allows for Complex Queries
  - MapReduce style filter and aggregations
  - Geospatial queries

P3.2 - Overview

- Build a social network about Reddit comments
- Dataset generated from Reddit.com
  - **users.csv**, **links.csv**, **posts.json**
- Build a social network timeline on the Reddit.com data
  - **Task 1**: Basic login
  - **Task 2**: Social graph
  - **Task 3**: Rank user comments
  - **Task 4**: Generate user timeline
  - **Task 5**: Caching mechanism
- **Task 6: Understanding Tail Latency, BLOBs, Storage Types, and Selecting Databases**
  - Answer questions on relevant topics and choose the right database and storage type for a given scenario

TDD* with Mockito

- Mockito is an open-source testing framework that allows the creation of test double objects (mock objects).
- It is used to mock interfaces so that the specific functionality of an application can be tested without using real resources such as databases, expensive API calls, etc.
- You are required to understand the given implementation, and may use it to quickly debug your solution for Task 1.

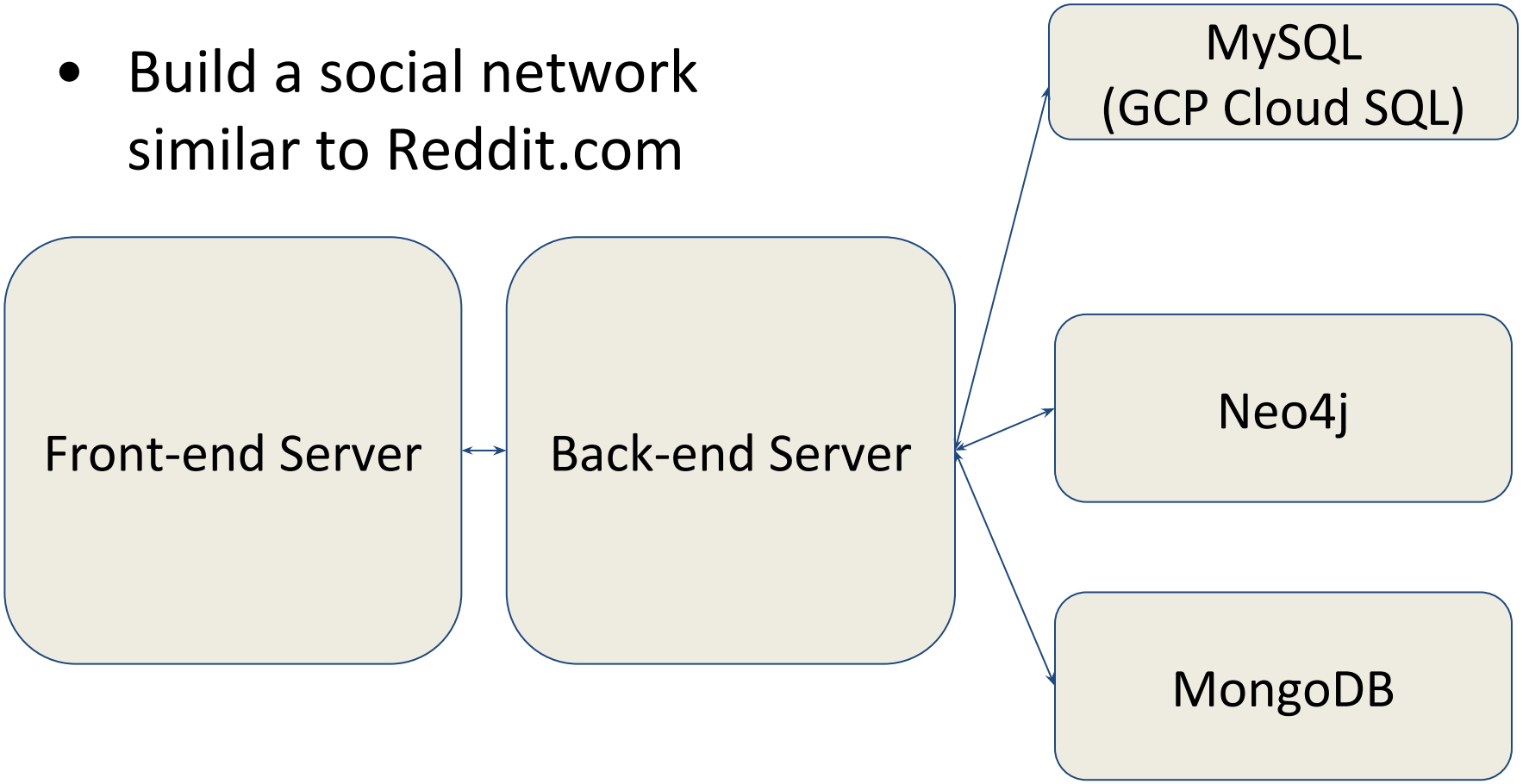
*Test Driven Development

P3.2 - Reddit Dataset

- Task 1: User profiles
  - User authentication system : GCP Cloud SQL(users.csv)
  - User info / profile : GCP Cloud SQL
- Task 2: Social graph of the users
  - Follower, followee : Neo4j (links.csv)
- Task 3: User activity system
  - All user generated comments : MongoDB (posts.json)
- Task 4: User timeline
  - Put everything together
- Task 5: Caching Mechanism
  - Cache the requests

P3.2 - Architecture

- Build a social network similar to Reddit.com



- Some images in the front-end are broken. No worries as long as you can get valid responses using “curl” command.

Tasks, Datasets & Storage

- Introduction
- The Scenario: Build Your Own Social Network Website
- Task 1: Implementing Basic Login with SQL
- Task 2: Storing Social Graph using Neo4j
- Task 3: Build Homepage using MongoDB
- Task 4: Put Everything Together
- Task 5: Caching Mechanism
- Task 6: Choosing Databases

Dataset Name	Data Store Type
Login Information	RDBMS
Relation	Graph Database
Comments	Document Stores
Profile Images	S3

P3.2 - Task 6

- **Issues of dealing with Scale**
  - An overview of the systems issues that arise with scale and how they were addressed in the context of Facebook.
    - Tail Latency and Fanout
    - BLOBs and Storage Types
      - Cost and performance
    - Learn how popularity and freshness of data plays a role in designing efficient social networking backends.

P3.2 - Task 6

- **Choosing Databases & Storage Types**
  - Use your knowledge and experience gained working with the databases in the project to
    - Identify advantages and disadvantages of various DBs
    - Pick suitable DBs for particular application requirements
    - Provide reasons on why a certain DB is suitable under the given constraints
  - Instructions provided in **runner.sh**

Terraform

- **Required in P3.2**
- **Required in the team project, get some practice**
- Use **‘terraform destroy’** to terminate resources
- This project is on GCP, so apply the following tag
  - The tag is “3-2” instead of “3.2” (for GCP only)

P3.2 - Reminders and Suggestions

- Set up a budget alarm on GCP
  - Suggested budget: $15
  - No penalties
- Learn and practice using a standard JSON Library. This will prove to be valuable in the Team Project
  - **Google GSON** - Recommended for Java
- Set up Gcloud in your environment
- No AWS instances on your individual AWS account are allowed
  - Otherwise you will receive warning emails and penalties

P3.2 - Reminders and Suggestions

- In Task 4 and 5, you will use the databases from all previous tasks. Make sure to have **all** the databases loaded and ready when working on Task 4 and 5.
- You can submit one task at a time using the submitter. Remember to have your Back-end Server VM running when submitting.
- Make sure to terminate **all** resources using “terraform destroy” after the final submission. Double check on the GCP console that all resources were terminated.

This Week's Deadlines

- Quiz 6: OLI Module 13

  Due: Friday, October 16th, 2020

- Project 3.2: Social Network Timeline with Heterogeneous backends

  Due: Sunday, October 18th, 2020 11:59PM ET

Q&A