15-319 / 15-619
Cloud Computing

Weekly Overview 6

Heterogeneous storage on the cloud

October 5, 2021

Overview

- **Recap of Last Week's activities**
  - OLI Unit 3 - Modules 7, 8, 9
  - Quiz 4
  - Project 2
  - OPE Training Session
  - Team Project, Phase 1, Microservice 1 Checkpoint
- **This week's activities**
  - Project 2 Discussion
  - OLI Unit 3 - Modules 10, 11, 12
  - Quiz 5
  - Project 3
  - Team Project, Phase 1, Microservice 1 Final
  - Team Project, Phase 1, Microservice 2 Checkpoint

Recap of Last Week's activities

- **OLI Unit 3: Virtualizing Resources for the Cloud**
  - Module 7: Introduction and Motivation
  - Module 8: Virtualization
  - Module 9: Resource Virtualization - CPU
- **Quiz 4**
- **Project 2, Containers: Docker and Kubernetes**
  - Docker and Kubernetes Intro / Embedded Profile Service deployment
  - Intro to Helm Charts / Deploying MySQL
  - WeCloud Chat Microservices
    - Autoscaling, Multi-Cloud and Fault-tolerance

Recap of Last Week's activities, cont.

- **OPE Training Session**
- **Team Project, Phase 1, Microservice 1 Checkpoint**

Value	Target RPS	Weight	Due date (at 11:59PM EST)
Microservice 1 Checkpoint	-	5%	Sunday, October 3rd
Checkpoint Report	-	5%	Sunday, October 3rd
Microservice 1 Early Bird Bonus	40,000	5%	Sunday, October 3rd

This Week's activities

- **Project 2 Discussion**
- **OLI 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: Heterogeneous Storage on the Cloud**
  - Introduction to MySQL
  - Introduction to NoSQL (HBase)

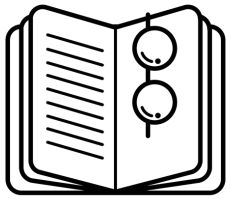
  Scenario: Build Your Own Social Network Website
  - Implementing Basic Login with SQL
  - Storing Social Relations as Graph using Neo4j
  - Build Homepage using MongoDB
  - Put Everything Together - Social Network Timeline
  - Caching

This Week's activities, cont.

- **Team Project, Phase 1, Microservice 1 Final**
- **Team Project, Phase 1, Microservice 2 Checkpoint**

Microservice 1 Final	40,000	10%	Sunday, October 10th
Microservice 2 Checkpoint	-	5%	Sunday, October 10th
Microservice 2 Correctness Bonus	-	Waive one most significant penalty for each team member	Sunday, October 10th
Microservice 2 Early Bird Bonus	25,000	5%	Sunday, October 10th

This Week: Conceptual Content



- OLI, UNIT 3: Virtualizing Resources for the Cloud
  - Module 7: Introduction and Motivation
  - Module 8: Virtualization
  - Module 9: Resource Virtualization - CPU
  - Module 10: Resource Virtualization - Memory
  - Module 11: Resource Virtualization – I/O
  - Module 12: Case Study
  - Module 13: Storage and Network Virtualization

OLI, Unit 3: Modules 10, 11, 12

- Understand two-level page mappings from virtual memory to real pages, from real pages to physical memory
- Learn how memory is overcommitted and reclaimed using ballooning
- Study how I/O requests are intercepted at different interfaces
- Map these concepts into your practical exploration with AWS

OLI Module 10 - Memory Virtualization

- A process that cannot fit into the physical memory? To run or not to run?
- Page Table
  - Per process
  - Maps virtual addresses to physical addresses
- One level vs. two levels mapping
- Virtual, Real, Physical address spaces
- Memory reclamation
- Ballooning

OLI Module 11 - I/O Virtualization

- How?
  - Construct a virtual version of the device
  - Virtualize the I/O activity routed to the device
- I/O Basics
- System call interface, device driver interface, and operation-level interface

OLI Module 12 - AWS Case Study

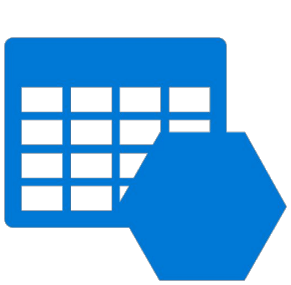
This Week: Project 3

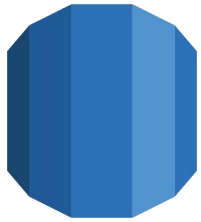
Heterogeneous Storage on the Cloud

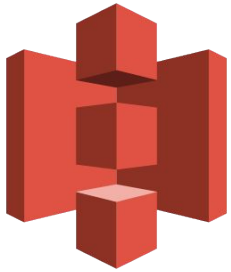
- Introduction to MySQL
- Introduction to NoSQL (HBase)

  Scenario: Build Your Own Social Network Website
- Implementing Basic Login with SQL
- Storing Social Relations as Graph using Neo4j
- Build Homepage using MongoDB
- Put Everything Together - Social Network Timeline
- Caching - Social Network Timeline with Cache

Project 3







Primers for Project 3

- MySQL Primer
- NoSQL Primer
- HBase basics
- MongoDB Primer

MySQL Primer

- **Introduction to *Structured Query Language* (SQL)**
  - Data Definition Language (DDL)
    - CREATE, ALTER, DROP
  - Data Manipulation Language (DML)
    - Create: INSERT, Read: SELECT, Update: UPDATE, Delete: DELETE
- **Table indexing**
  - Single column vs Multi-column indexing
  - Common pitfalls
- **Storage Engines**
  - MyISAM
  - InnoDB

Storage Engines in MySQL

- A storage engine is a software module that a DMS uses to create, read, update data from a database
- **MyISAM** and **InnoDB**
- They have:
  - Different caching mechanisms
  - Different locking mechanisms
  - Are optimized for either read or write
  - More differences are explained in the primer

Experiment, and think of which one to use in the team project

Read the MySQL primer

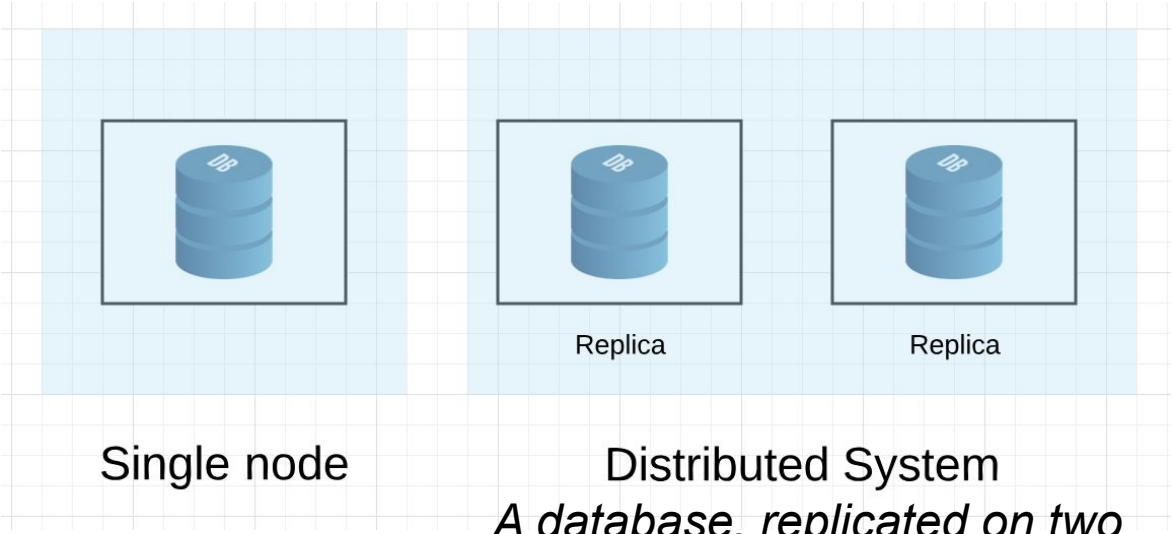
NoSQL Primer

- **Non-SQL or NotOnly-SQL**
  - Non-relational
- **Why NoSQL if we already have SQL solutions?**
  - Flexible data model (schemaless, can change)
  - Designed to be distributed (scale horizontally)
  - Certain applications require improved performance at the cost of reduced data consistency (data staleness)
- **Basic Types of NoSQL Databases**
  - Schema-less Key-Value Stores (Redis)
  - Wide Column Stores (Column Family Stores) (HBase)
  - Document Stores (MongoDB)
  - Graph DBMS (Neo4j)

CAP Theorem

- It is impossible for a distributed data store to provide all the following three guarantees at the same time:
  - **Consistency:** no stale data
  - **Availability:** no downtime
  - **Partition Tolerance:** network failure tolerance in a distributed system

Single Node to Distributed Databases

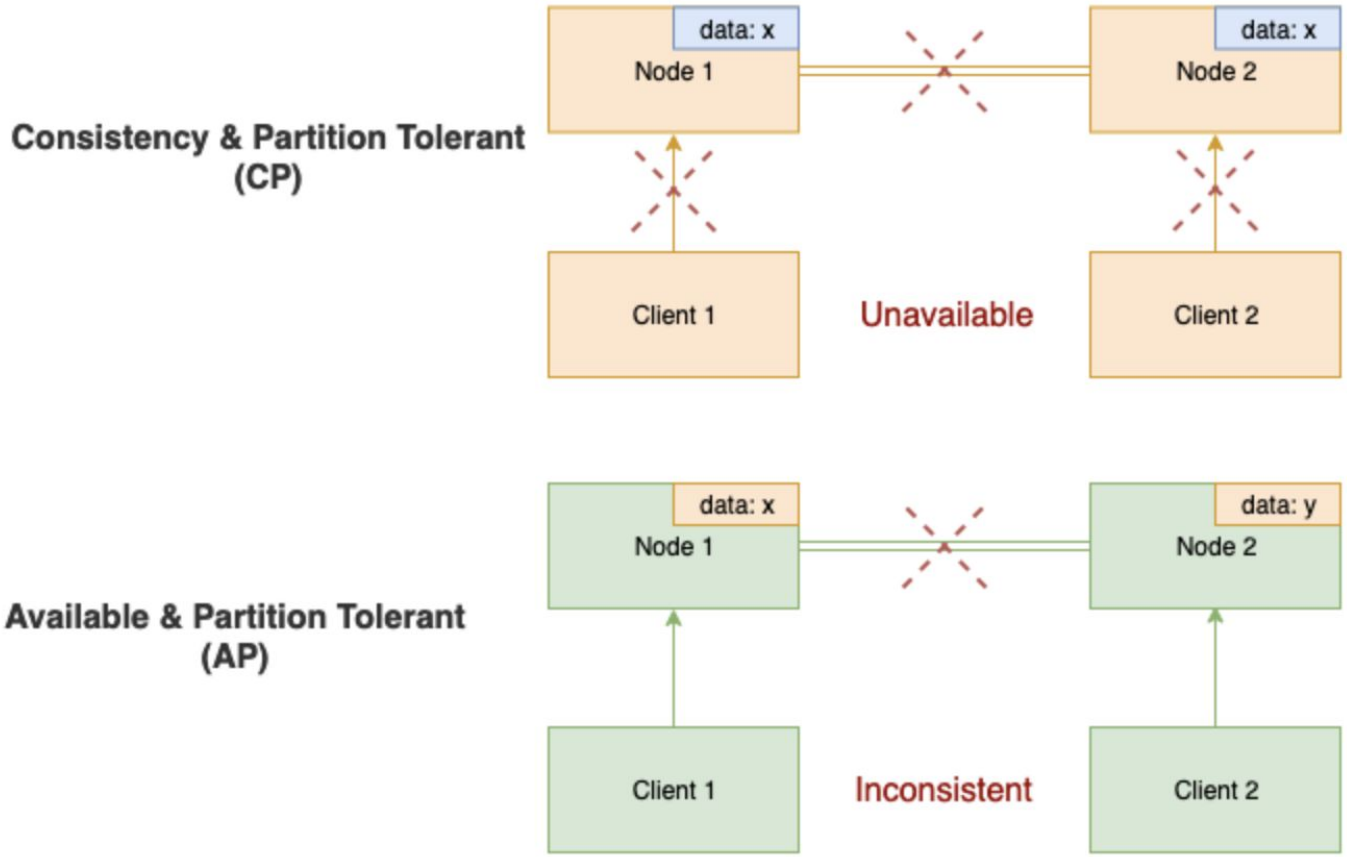


- Since DB is replicated, how is consistency maintained?
- Since the data is replicated, if one replica goes down, will the entire service go down?
- How will the service behave during a network failure?

CAP Theorem

- Only two out of the three are feasible:
  - **CA: non-distributed (MySQL, PostgreSQL)**
    - Traditional databases like MySQL and PostgresQL have only one server
    - Don’t provide partition tolerance
  - **CP: downtime (HBase, MongoDB)**
    - Stop responding if there is partition
    - There will be downtime
  - **AP: stale data (Amazon DynamoDB)**
    - Always available
    - Data may be inconsistent among nodes if there is a partition

Only two at a time



HBase basics

- **Introduction to HBase**
  - **HBase Operations**
  - **HBase Architecture**
- **HBase Tutorial**
  - Set up standalone HBase
  - Creating the HBase Cluster with AWS EMR
  - Creating the HBase Cluster with Azure HDInsight
  - Loading data into HBase
  - HBase Compaction
  - Compression and Data Block Encoding in HBase
  - HBase Data Migration
- **HBase Query**
- **HBase Java API**
- **Security Best Practice of Hadoop clusters**

MongoDB Primer

- **Compare MongoDB and MySQL**
- **MongoDB Features**
- **MongoDB Technicalities**
  - Documents
  - Collections
- **MongoDB Tutorial to practice:**
  - How to import data into MongoDB
  - Some basic queries with Mongo Shell
  - How to Build index to speed up your query

Project 3 Overview

- Task 1: Introduction to MySQL
  - Load data, run queries, indexing
  - Plain-SQL vs ORM
- Task 2: Introduction to NoSQL (HBase)
  - Load data, design key, run basic queries

Scenario: Build Your Own Social Network Website using datasets from Reddit.com: **users.csv**, **links.csv**, **posts.json**

- Task 3: Implementing Basic Login with SQL
  - User authentication system : Azure Database for MySQL(users.csv)
  - User info / profile : Azure Database for MySQL
- Task 4: Storing Social Graph using Neo4j
  - Follower, followee : Neo4j (links.csv)

Project 3 Overview, cont.

- Task 5: Build Homepage using MongoDB
  - All user generated comments: MongoDB (posts.json)
- Task 6: Put Everything Together
  - User Timeline: Fanout
- Task 7: Caching
  - Cache the requests with high frequency

Task 1: Introduction to MySQL

- Prepare tables
  - A script to create the table and load data is provided
- Write MySQL queries to answer questions
- Learn JDBC
- Complete MySQLTasks.java
- Aggregate functions, joins
- *Statement* and *PreparedStatement*
- SQL injection
- Learn how to use proper indexes to improve performance

Dataset for Task 1

https://www.yelp.com/dataset_challenge

- business
- checkin
- review
- tip
- user

MySQL Indexing

- **Schema**
  - The structure of the tables and the relations between tables
  - Based on the structure of the data and the application requirements
- **Index**
  - An index is simply a pointer to data in a table
  - It is a data structure (lookup table) that helps speed up the retrieval of data from tables (e.g., B-Tree, Hash indexes, etc.)
  - Based on the data as well as queries
  - Build indexes based on the types of queries you'll expect

We have an insightful section about the practice of indexing, read it carefully! **Very helpful for the team project**

EXPLAIN statement in MySQL

- **How do we evaluate the performance of a query?**
  - Run it
- **What if we want/need to predict the performance without execution?**
  - Use EXPLAIN statement
- **The EXPLAIN statement on a query predicts:**
  - The number of rows to scan
  - Whether it makes use of indexes or not

Object Relational Mapping (ORM)

- **ORM abstracts the interaction with a DB for you:**
  - Maps the domain class with the database table
  - Map each field of the domain class with a column of the table
  - Map instances of the classes (objects) with rows in the corresponding tables

	Mapped to	
public class Course {	→	course
String courseId;	→	course_id (PK)
String name;	→	name
}		
Domain Class	→	Database Table
Objects	→	Rows

Benefits of ORM

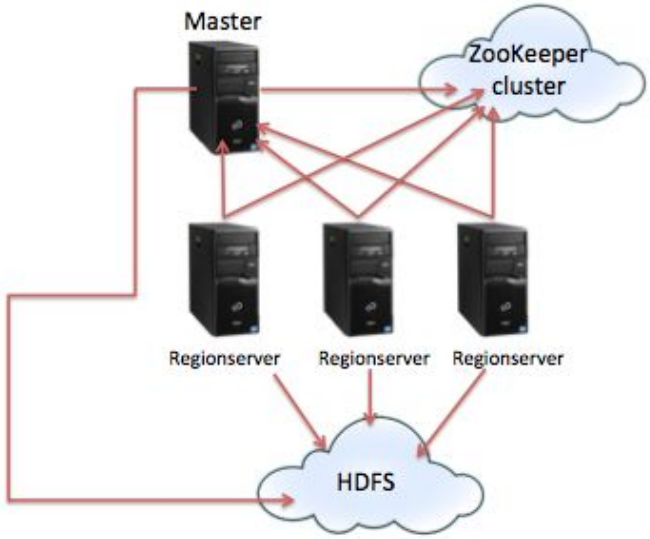
- **Decoupling of responsibilities**
  - ORM decouples the CRUD operations and the business logic code
- **Flexibility to meet evolving business requirements**
  - Cannot eliminate the schema update problem, but it may ease the difficulty, especially when used together with data migration tools
- **Persistence transparency**
  - Changes to a persistent object will be automatically propagated to the database without explicit SQL queries
- **Productivity**
  - No need to keep switching between your OOP language such as Java/Python, etc. and SQL
- **Vendor independence**
  - Abstracts the application from the underlying SQL database and SQL dialect

ORM Question in the MySQL Task

- The current business application exposes an API that returns the most popular Pittsburgh businesses
- It is based on a SQLite3 database with an outdated schema
- **Your task:**
  - Plug the business application to the MySQL database and update the definition of the domain class to match the new schema
- The API will be backwards compatible without modifying any business logic code

Task 2: Introduction to NoSQL (HBase)

- HBase is an open source, column-oriented, distributed database developed as part of the Apache Hadoop project



In this task, you will:

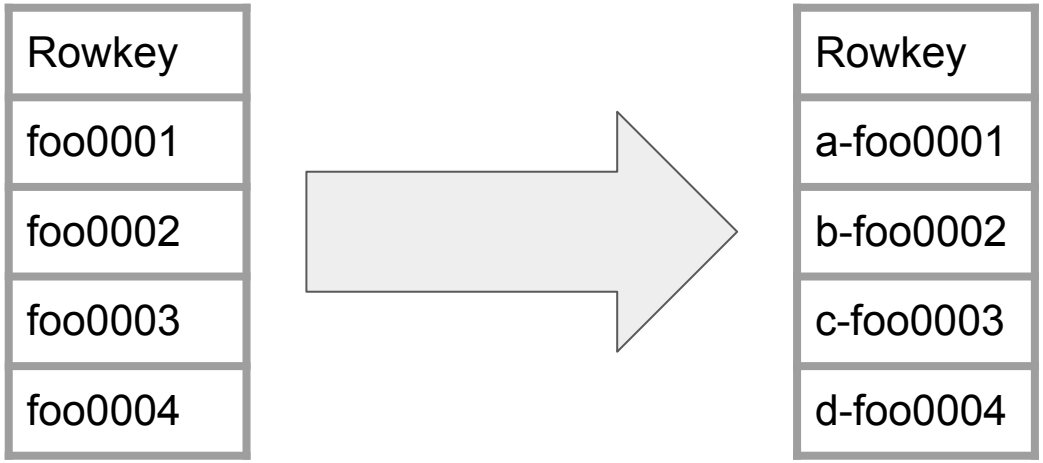
- Launch an HDInsight cluster
- Load data so that it is evenly distributed across regions
  - **Make sure to submit a *design.pdf* file with your key design**
- Try different commands in the *hbase-shell*
- Complete *HBaseTasks.java* using HBase Java APIs

RowKey Design

- Rows in HBase are sorted lexicographically by rowkey
- **Hotspotting**
  - A large amount of client traffic is directed to one/few node/s
  - The rows are divided into different HRegions
  - Each HRegion contains a contiguous subset of rows
  - HRegionServer is responsible for reading and writing
  - Solution: pre-splitting regions

RowKey Design - Example 1

Salting: randomly assign prefix



RowKey Design - Example 1

Salting: randomly assign prefix

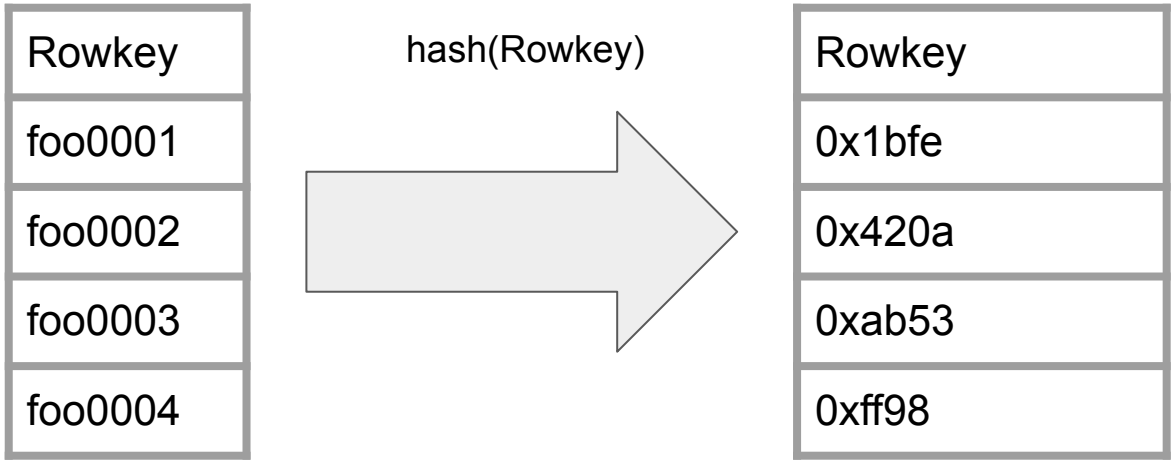
- Command in HBase shell

  > create 'table', ‘example1’', SPLITS=> ['b', 'c', 'd’']

Region	Start Key	End Key	Rows
1		b	a-foo0001
2	b	c	b-foo0002
3	c	d	c-foo0003
4	d		d-foo0004

RowKey Design - Example 2

Hashing



RowKey Design - Example 2

Hashing

- Command in HBase shell

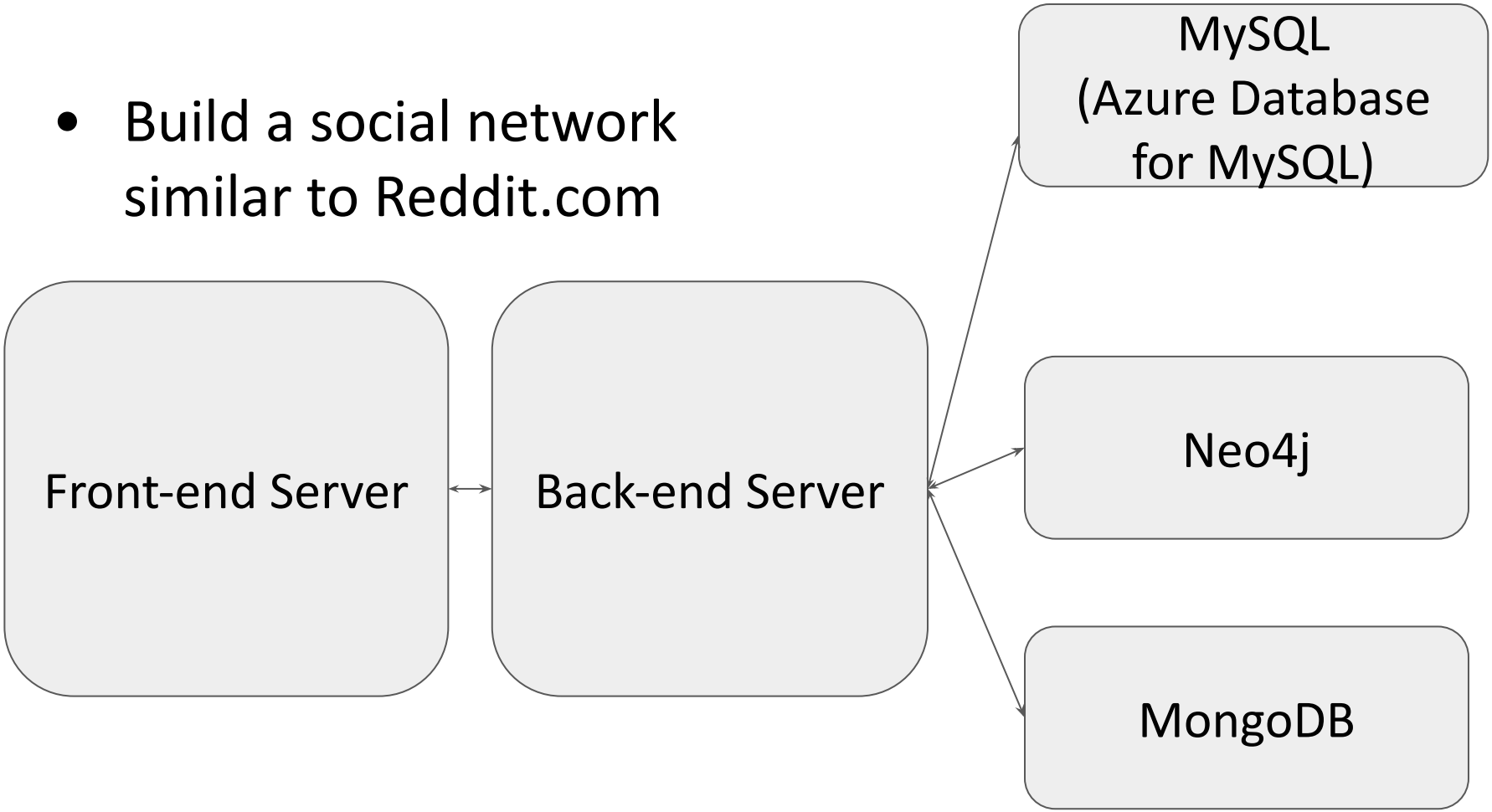
  > create 'table', 'example2', {NUMREGIONS => 4, SPLITALGO => 'HexStringSplit'}

Region	Start Key	End Key	Rows
1		3fff	0x1bfe
2	3fff	7fff	0x420a
3	7fff	bfff	0xab53
4	bfff		0xff98

Task 3 - Task 7: A Social Network Service

Social Network 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.

Task 3: Implementing Basic Login with MySQL

- Designed to managed highly structured data.
  - Authentication data

- Database-as-a-Service (DBaaS)
  - Azure-managed MySQL database
    - Cloud vendor is responsible for administrative tasks
    - Users are responsible for optimizing applications that use database resources

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.

Task 4: Storing Social Graph using 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

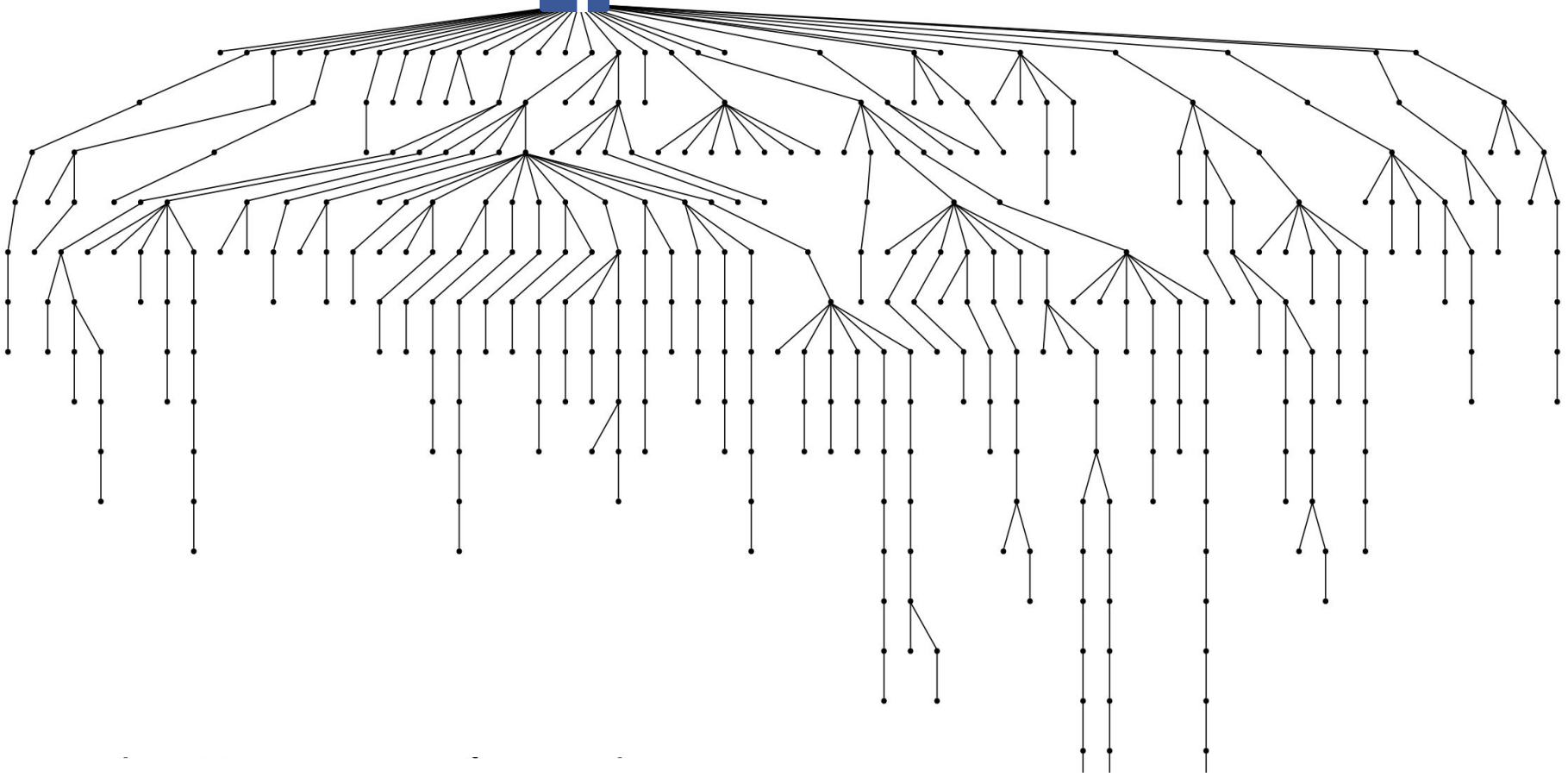
Task 5: Build Homepage using MongoDB

- 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

Task 6: Social Network Timeline

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).

Task 6: Social Network Timeline
High Fanout in Data Fetching

- Practice writing complex fan-out queries that span multiple databases:
  - MySQL
  - Neo4J
  - MongoDB

Task 7: Social Network Timeline with Cache

- Fanout and Caching
  - Practice writing complex fan-out queries that span multiple databases.
  - Also practice using a caching mechanism to boost your backend!

P3 - Tagging and Budget limit

- This project is on Azure, so apply the following tag appropriately:

Tags Required	Key: `project` Value: `cloud-storage`

- The project does not have a hard budget limit. Each student is assigned a single subscription for all the projects. That being said, the students should still be careful of the spending since the subscription can get deactivated if they spend up the entire budget, particularly when working with HDInsight clusters.

P3 - Reminders and Suggestions

- Students are encouraged to work on the Intro to DB (Task 1 & 2) first, and then Social Network. The Social Network will require prior knowledge/experience on Task 1 & 2.
- We discourage installing VSCode Java Language Server on the VM because it could add heavy loads on the VM's CPU and memory which might then cause a OOM failure when loading large-scale raw data to the databases. Instead, develop locally, scp to and test on the VM.
- Use tmux to prevent session timeout which could immensely enhance the development experience.

P3 - Reminders and Suggestions

- In Task 6 and 7, you will use the databases from tasks 3 to 5. Make sure to have **all** the databases loaded and ready when working on Task 6 and 7.
- 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 Azure console that all resources were terminated.
- No AWS instances on your individual AWS account are allowed, otherwise you will receive warning emails and penalties

P3 - Reminders and Suggestions, contd.

- **Tag your resources:**
  - **Key: project, Value: cloud-storage**
- An HDInsight cluster is very expensive
  - Exercise caution to plan for the budget
- Be patient:
  - Provisioning an HDInsight cluster takes ~30min
  - Loading data to MySQL takes ~40 minutes
  - Loading data to MongoDB takes ~30 minutes

  *Project 3 is due in two weeks but **please start early** because of the above wait times.*
- **Remember to delete the Azure resource group to clean up all the resources in the end. If you leave an HDInsight cluster running and exceed the budget of the subscription, you will be unable to work on the future Azure projects.**

Upcoming Deadlines

- Project 2 Discussion
  - Due next Sunday, 2021-10-10 23:59 ET
- Quiz 5
  - Due on Friday, 2021-10-08 23:59 ET
- Team Project, Phase 1
  - Microservice 1 Final
  - Microservice 2 Checkpoint
  - Microservice 2 Correctness Bonus
  - Microservice 2 Early Bird Bonus
    - Due next Sunday, 2021-10-10 23:59 ET