15-319 / 15-619
Cloud Computing

Weekly Overview 6

Cloud Storage - SQL and NoSQL

Team Project

February 22, 2022

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 Released
- **This week's activities**
  - Project 2 Discussion
  - OLI Unit 3 - Modules 10, 11, 12
  - Quiz 5
  - Project 3 - Part 1 - SQL and NoSQL; Reflection
  - Team Project, Phase 1, Microservice 1 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

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 - Part 1: Cloud Storage - SQL and NoSQL**
  - Introduction to MySQL
  - Introduction to NoSQL (HBase)

Project 3 - Two Parts

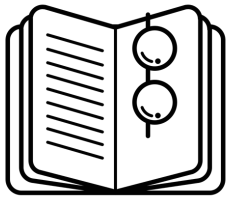
- **Project 3 - Part 1: Cloud Storage - SQL and NoSQL**
  - Open: 21th February 00:00
  - **Due: 27th February 11:59 PM ET**
  - Duration: 1 week
  - Grades: 50% of P3
- **Project 3 - Part 2:**
  - Open: 28th February 00:00
  - **Due: 6th March 11:59 PM ET**
  - Duration: 1 week
  - Grades: 50% of P3

This Week's activities, cont.

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

Microservice 1 Checkpoint	-	5%	Sunday, February 27th
Checkpoint Report and & Team Introduction survey	-	5%	Sunday, February 27th
Microservice 1 Early Bird Bonus	65,000	5%	Sunday, February 27th

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

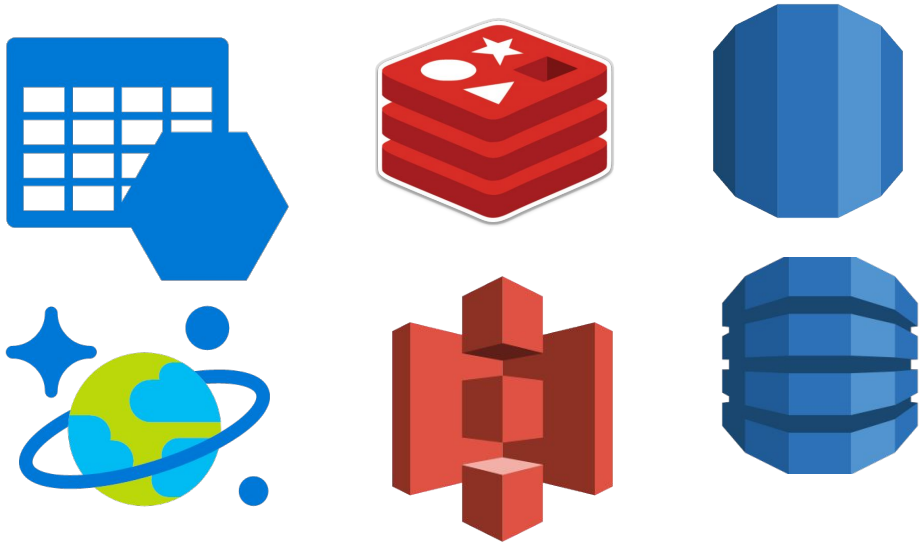
- A process that cannot fit into the physical memory? To run or not to run?
- Page Table
  - Per process
  - Maps virtual addresses to physical addresses
- One level vs. two levels mapping
- Virtual, Real, Physical address spaces
- Memory reclamation
- Ballooning

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

Project 3



Project 3 Special Deadline & Grading Policies

- Two Parts: SQL & NoSQL, Heterogeneous storage on the cloud; **ONE WEEK** to finish each part
  - By Sunday, February 27th at 11:59 PM
    - SQL and NoSQL -- Complete the two tasks and make a reflection post
  - By Sunday, March 6th at 11:59 PM
    - SQL and NoSQL -- Complete discussion tasks
    - Heterogeneous Storage on the Cloud -- Complete the tasks and make a reflection post
  - By Sunday, March 13th at 11:59 PM
    - Heterogeneous Storage on the Cloud - Finish discussion tasks

Project 3 Special Deadline & Grading Policies (Cont.)

- Grading: Each of the two parts is worth 100 points and contributes to 50% of P3 final grade. If you skip any one of the two parts, you will only get **at most 50%** for the final grade of P3.

Project 3 SQL & NoSQL 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

Primers for Project 3 SQL and NoSQL

- MySQL Primer
- NoSQL Primer
- HBase Basics

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

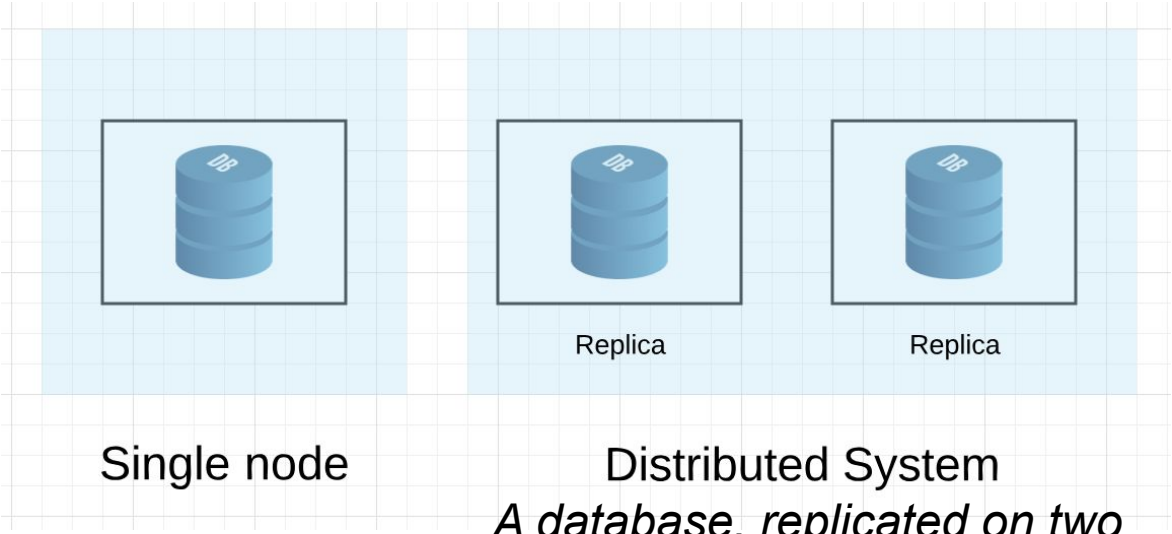
NoSQL Primer

- **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

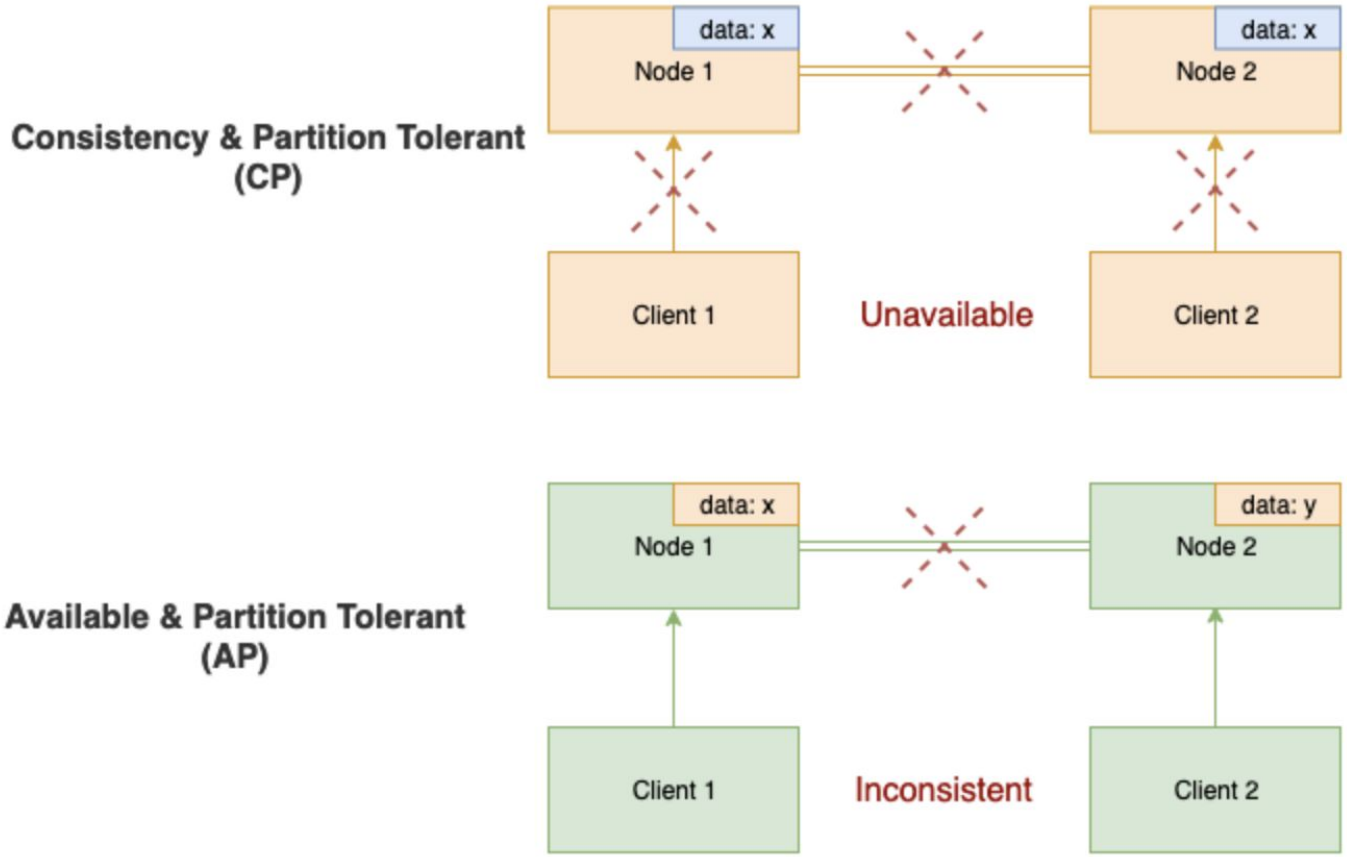


- Since DB is replicated, how is consistency maintained?
- Since the data is replicated, if one replica goes down, will the entire service go down?
- How will the service behave during a network failure?

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**

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

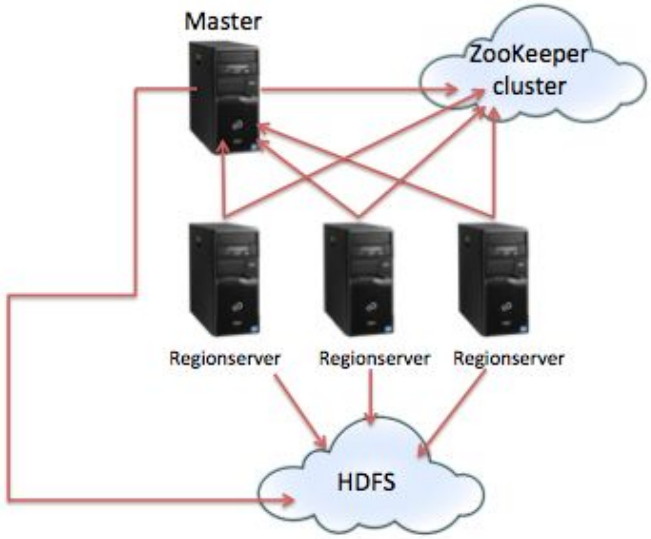
- **Decoupling of responsibilities**
  - ORM decouples the CRUD operations and the business logic code
- **Flexibility to meet evolving business requirements**
  - Cannot eliminate the schema update problem, but it may ease the difficulty, especially when used together with data migration tools
- **Persistence 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:

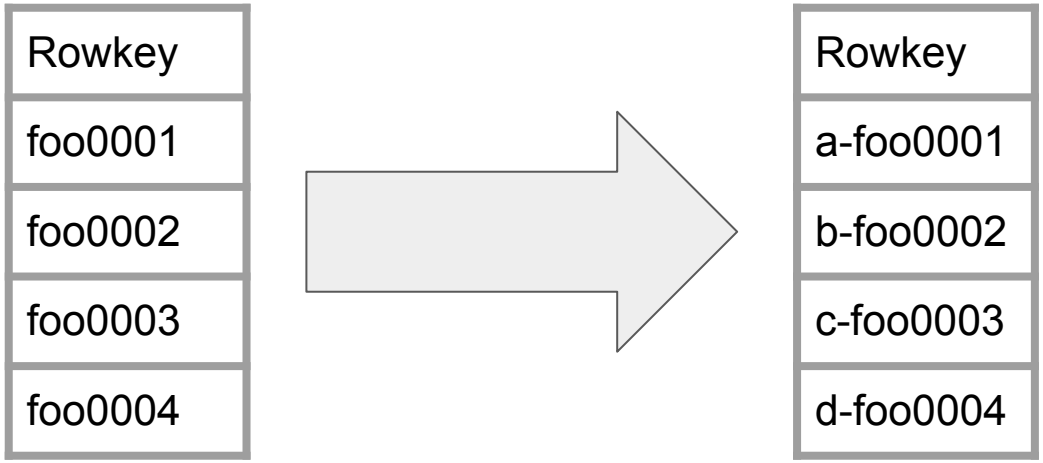
- Launch an HDInsight cluster
- Load data so that it is evenly distributed across regions
  - **Make sure to submit a *design.pdf* file with your key design**
- Try different commands in the *hbase-shell*
- Complete *HBaseTasks.java* using HBase Java APIs

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

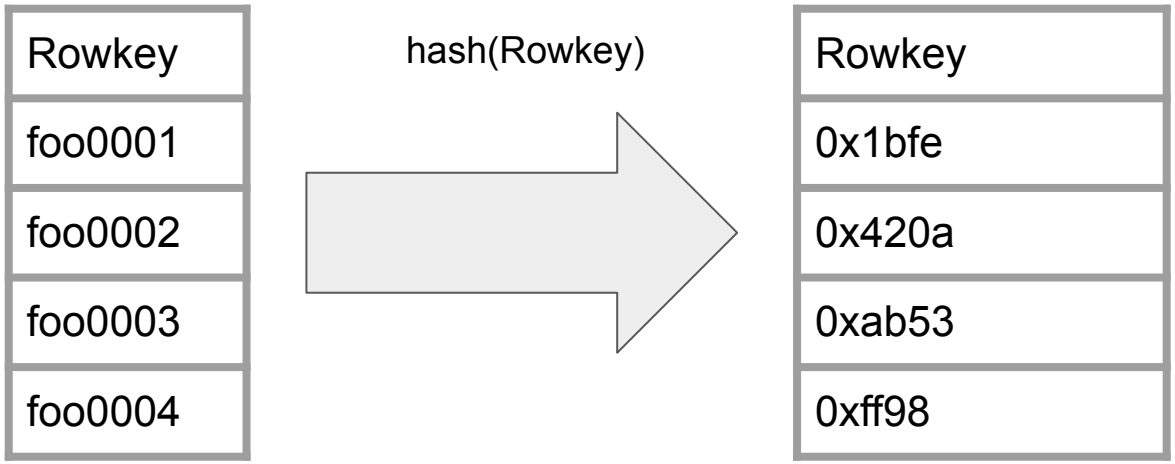
- Command in HBase shell

  > 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

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

- 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.

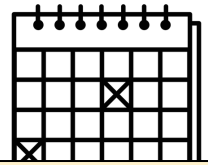
P3 - Reminders and Suggestions, cont.

- Be patient:
  - Provisioning an HDInsight cluster takes ~30min
  - Loading data to MySQL takes ~40 minutes

  *You only have **one week** for SQL and NoSQL , so **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.**

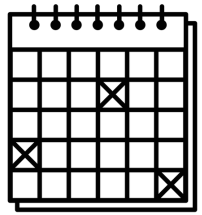
TEAM PROJECT
Twitter Data Analytics

Team Project Time Table



Phase	Deadline (11:59PM ET)
Phase 1 (20%) - **M1** - **M2** - **M3 (ckpt)**	• **M1 CKPT (5%): Sun, 2/27** • **M1 CKPT Report (5%) + Team Intro Form: Sun, 2/27** • M1 FINAL (10%): Sun, 3/6 • M2 CKPT (5%): Sun, 3/6 • M2 FINAL (50%): Sun, 3/20 • M3 CKPT (5%): Sun, 3/20 • Final Report + Code (20%): Tue, 3/22 **BONUSES:** • **M1 Early Bird Bonus (5%): Sun, 2/27** • M2 Early Bird Bonus (5%): Sun, 3/6 • M2 Correctness Penalty Waiver: Sun, 3/6 • M3 Early Bird Bonus (5%): Sun, 3/20 • M3 Correctness Penalty Waiver: Sun, 3/20

Suggested Tasks for Phase 1



Phase 1 weeks	Tasks	Deadline
Week 1-2 ● **02/14 - 02/27**	● **Team meeting** ● **Read Write Up & Report** ● **Complete M1 code & achieve correctness** ● **Start writing M2 solution** ● **Think about M3 database schema**	● **M1 Checkpoint due on 02/27** ● **Checkpoint Report due on 02/27**
Week 3 ● 02/28 - 03/06	● Optimize for M1 performance ● Complete correct M2 code ● **Start ETL process for M3**	● M1 final target due on 03/06 ● M2 Checkpoint due on 03/06
Week 4-5 ● 03/07 - 03/20	● Optimize for M2 performance ● Finish M3 ETL process ● Complete M3 code & achieve correctness	● M2 final target due on 03/20 ● M3 Checkpoint due on 03/20 ● Final Report due on 03/22

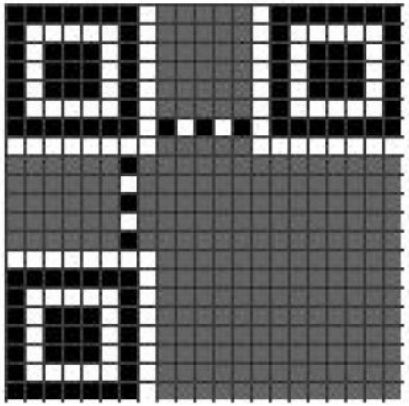
Microservice 1 - QR code

Submission Budget: $0.70/h

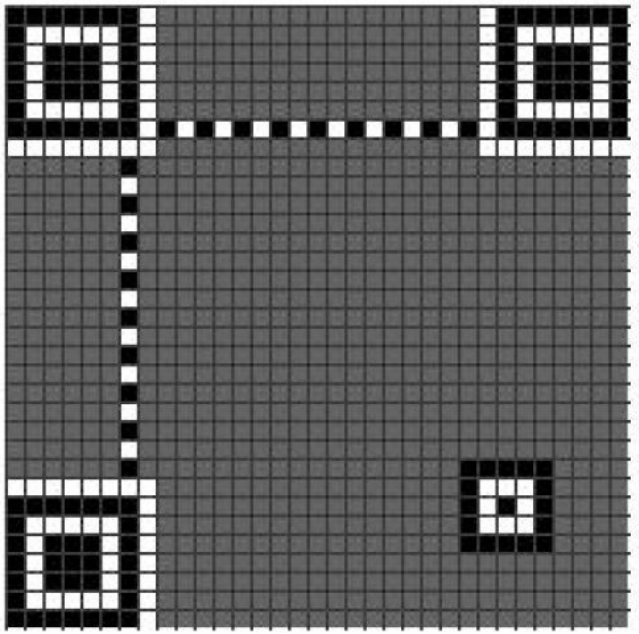
Target RPS: 65000

- M1 does not require a database
  - Implement encoding and decoding of QR code
    - A simplified version of QR
  - You must explore different web frameworks
    - Get at least 2 different web frameworks working
    - Techempower will be a good resource to explore
    - Select the framework with the better performance
    - Provide evidence of your experimentation

QR Code

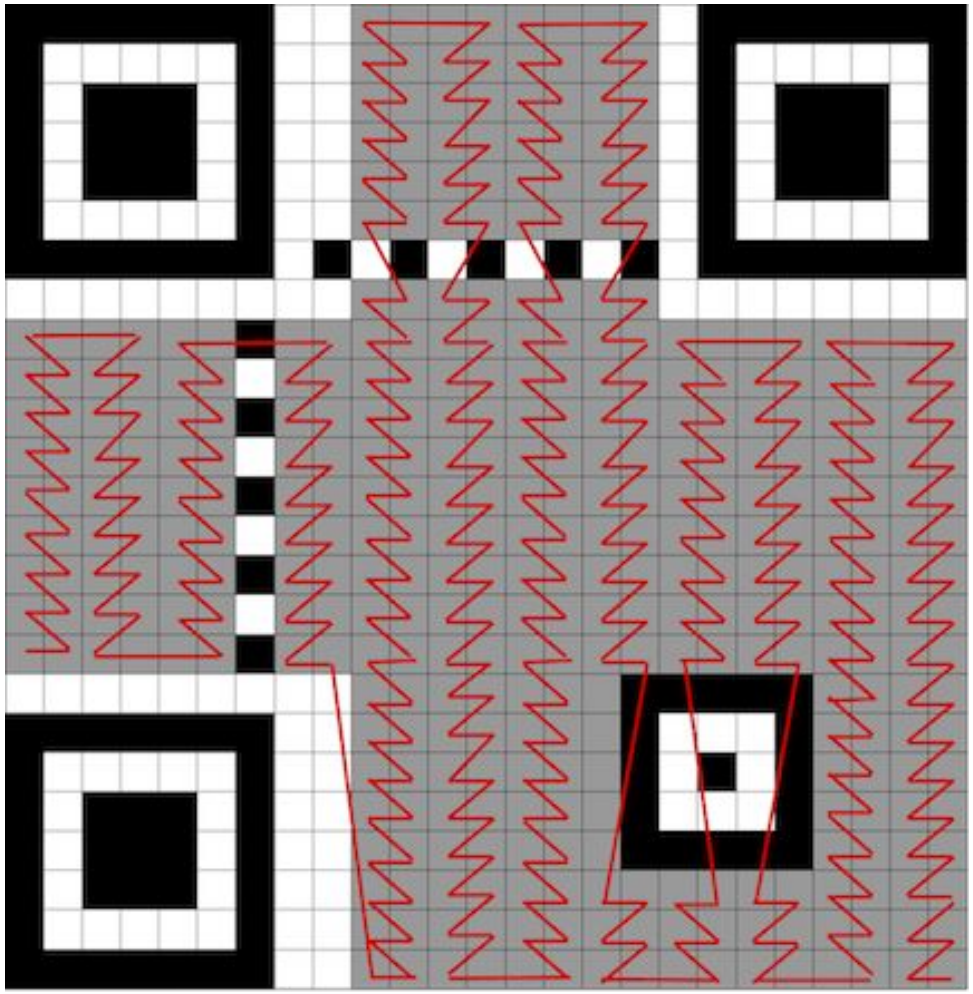


Version 1 (21 * 21)



Version 2 (25 * 25)

QR Code



Microservice 1 Tips

- Start with a single EC2 instance, work with cluster only when you are confident
- My RPS is low
  - Does your program utilizes all CPU core? Make sure threads / workers are set up properly.
  - Consider profiling. The Primer can be useful.
  - Try out different instance types. m6g can have better performance/cost ratio.
  - Try different framework or even language.
- After deploying to cluster, make sure the workload is evenly distributed across your worker nodes
- Check report for hints!

How to help TA to help you

- Hint: hint won't come from nowhere.
  - The more context you provide, the more we can understand your situation, the more accurate help we can offer
- Generate context: see Piazza guideline post
- Context for asking correctness improvement:
  - A checklist, flowchart, mind map describing your understanding
  - Measurements you've taken to check each item in the checklist/flowchart/mind map
    - For example, "I've wrote unit tests for everything in my checklist"

Reminders on penalties

- Self-managed Kubernetes cluster + optional EMR, consisting of 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 final 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 $1.10/hour (HBase)** applies to every submission

Upcoming Deadlines

- Quiz 5
  - Due on Friday, 2022-02-25 23:59 ET
- Project 2 Discussion
  - Due this Sunday, 2022-02-27 23:59 ET
- Project 3 - Part 1 - SQL and NoSQL Tasks and Reflection
  - Due this Sunday, 2022-02-27 23:59 ET
- Team Project, Phase 1
  - Microservice 1 Checkpoint
  - Checkpoint Report and & Team Introduction survey
    - Due this Sunday, 2022-02-27 23:59 ET

Best Wishes!!!