15-319 / 15-619
Cloud Computing

Recitation 6

P2.3 and P3.1

October 1, 2019

Overview

- **Last week’s reflection**
  - OLI Unit 3, Module 7, 8, 9
  - Quiz 4
- **This week’s schedule**
  - OLI Unit 3 - Module 10, 11, 12
  - Quiz 5
  - Project 2.3
  - Project 3.1 (Optional)
    - P3.1 is a pre-requisite for the Team Project

Last Week

- **Unit 3: Virtualizing Resources for the Cloud**
  - Module 7: Introduction and Motivation
  - Module 8: Virtualization
  - Module 9: Resource Virtualization - CPU
- **Quiz 4**
- **Project 2.2, Containers: Docker and Kubernetes**
  - Docker Intro / Embedded Profile Service
  - Intro to Helm Charts / Deploying MySQL
  - WeCloud Chat Microservices / Connected Architecture
    - Autoscaling, Multi-Cloud and Fault Tolerance to Azure

This Week

- **Unit 3: Virtualizing Resources for the Cloud**
  - Module 10: Resource virtualization (Memory)
  - Module 11: Resource virtualization (I/O devices)
  - Module 12: Case Study
- **Quiz 5**
- **Project 2.3, Functions as a Service (FaaS)**
  - Task 1, Explore functions on various CSPs
    - Azure Functions, GCP Cloud Functions, AWS Lambda
  - Task 2, Extract thumbnails from video stream
    - Azure Functions and FFmpeg
  - Task 3, Get image labels and index
    - Azure Computer Vision, Azure Search

Also This Week

- **Project 3.1, Files v/s Databases [Optional]**
  - Flat files
  - MySQL
  - Redis and Memcached
  - HBase
- **Sign up for the OPE by Saturday 10/05**
- **Make sure your teams are formed on TPZ by Friday 10/04**

Team Project - Time to Team Up

15-619 Students:

- Start to form your teams
  - Choose carefully as you cannot change teams
  - Look for a mix of skills in the team
    - Web tier: web framework performance
    - Storage tier: deploy and optimize MySQL and HBase
    - Extract, Transform and Load (ETL)

- Create a new AWS account only for the team project

15-319 Students:

- You are allowed to participate in the team project
- Once committed to a team, you cannot quit
- Earn a significant bonus for participating in the team project
- If you are a 15-319 student and want to participate in the team project, please email the professor.

Team Formation - Deadlines

Follow the instructions in @913 carefully

- **By Friday 10/04 at 11:59 PM ET**
    - Identify your team members
    - One team member should form a team on TPZ and all other team members should accept the invitation
        - Completing this step will freeze your team

- **By Saturday 10/05 at 11:59 PM ET**
    - Create a new AWS account ⇒ only used for the team project
    - Update the team profile in TPZ with the
        - New AWS ID aws-id

- **By Sunday 10/06 at 11:59 PM ET**
    - Finish reading the Profiling a Cloud Service primer to get yourself prepared for the team project

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 in physical memory? To run or not to run?
- Page Table
    - Per process
    - Maps virtual addresses to physical addresses
- One level v/s 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
- System call interface, the device driver interface, and the operation-level interface

OLI Module 12 - AWS Case Study

This Week’s Projects

Project 2: Automating and Autoscaling Distributed Services

- P2.3: Functions as a Service
  - Building a serverless application using Azure Functions

Project 3: Storage and DBs on the cloud

- P3.1: Files v/s Databases (Optional)
  - Comparison and usage of Flat files, RDBMS (MySQL) and NoSQL (Redis, HBase)

Project 2

Running Theme:
Automating and scaling distributed systems

P2.1
EC2
VMs

P2.2
Containers

P2.3
Functions

This week!

Serverless Computing

- Develop and run applications on servers without having to provision or manage servers.
- Applications can have one or more functions.
- The Cloud Service Provider (CSP) provides the server to run the application.
- The developer designs the application and runs it without having to manage any servers or worry about scaling.
    - Pay-per-invocation model
- Functions-as-a-Service (FaaS) is a use-case of serverless computing

Why use Cloud Functions

- High availability
- Resiliency
  - Each execution is contained and isolated, and thus has no impact on other executions
- Built in logging and monitoring
  - Easily accessible logs ensure traceability
- Event-driven
  - Functions can be triggered by events like Blob storage file uploads

Cloud Functions

- Possible use cases
  - Chat bots
  - Mobile backends
  - How about Mapreduce?
    - Not suitable for FaaS
    - EMR is PaaS
- FaaS, typically stateless functions
- Pay per number of function invocations + running time
- Scalability is automatic through provider

Google Cloud Functions

Triggers

- Events trigger functions
- Events are passed to function as input parameters
- Event sources publish events that cause the cloud function to be invoked
    - Azure: HTTPTrigger, BLOBTrigger, Event Grid etc.
    - AWS Lambda: Create object in S3 bucket, SNS topic etc.
    - GCP: File upload to Cloud Storage, message on a Cloud Pub/Sub topic etc.

Azure Functions

- Azure FaaS offering
- Only pay for the number of invocations and the compute time (i.e., when your code runs)
- Stateless…
  - No data is persisted across invocations
  - Can use persistent databases to maintain state
- Debugging?
  - Use Application Insights

Azure Function Model

- Triggers and events (HTTP Triggers, Blob Storage Triggers etc.)
- Context
  - Allows function to interact with Azure Function execution environment e.g. to get function name, function directory, logger etc.
- Logging and Monitoring
- Be stateless!

Project 2.3 - Overview

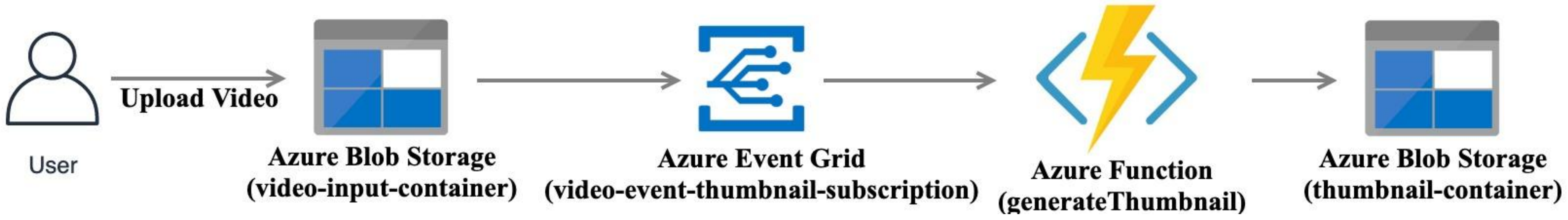
- Task 1
  - Fibonacci function in Azure Functions
  - Power sets function in GCP Cloud Functions
  - CIDR block statistics in AWS Lambda
- Task 2
  - Azure Functions, Azure Event Grid and FFmpeg to generate thumbnails
- Task 3
  - Azure Cognitive Computer Vision and Azure Search to label thumbnails and index for video search

P2.3 Task 1: Cloud Functions

- HTTP triggered functions
- Subtask 1
  - Fibonacci - Azure Functions
- Subtask 2
  - PowerSets - GCP Functions
- Subtask 3
  - CIDR - AWS Lambda
    - Java: use SubnetUtils class from Apache's Commons Net.
    - Python users should consider the python-iptools package or ipaddress from the Python standard library.

P2.3 Task 2: Event Driven Functions

- Goal: Build a serverless pipeline using Event driven functions and Azure Event Grid
- What is Event Grid?
  - An event delivery mechanism which allows Functions to subscribe to event topics.
  - Provides a reliable event delivery and event filtering functionality.

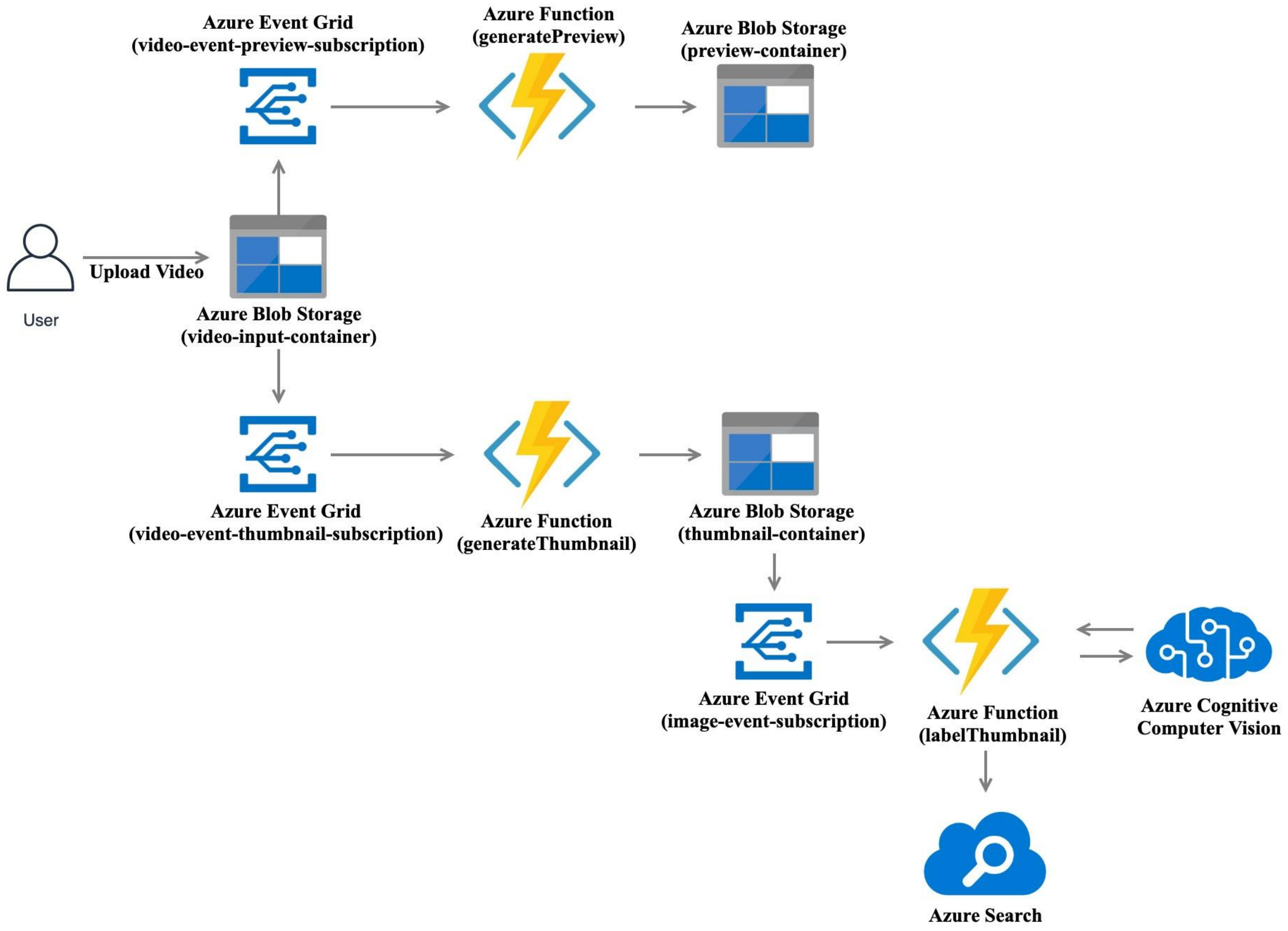

User
Upload Video
Azure Blob Storage (video-input-container)
Azure Event Grid (video-event-thumbnail-subscription)
Azure Function (generateThumbnail)
Azure Blob Storage (thumbnail-container)

Project 2.3 - Task 3

Goal: Build a Video Processing Pipeline

- Azure Functions and FFmpeg for processing videos
- Azure Cognitive Computer Vision for labeling thumbnails
- Azure Search to index videos based on thumbnail labels

Project 2.3 - Task 3

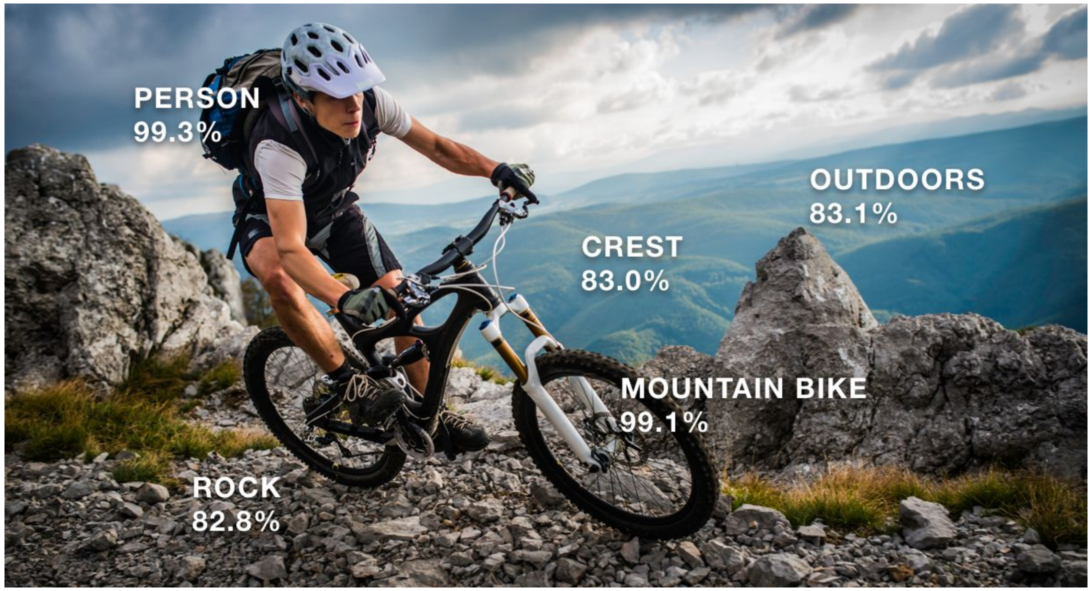

Azure Event Grid (video-event-preview-subscription)
Azure Function (generatePreview)
Azure Blob Storage (preview-container)
User
Upload Video
Azure Blob Storage (video-input-container)
Azure Event Grid (video-event-thumbnail-subscription)
Azure Function (generateThumbnail)
Azure Blob Storage (thumbnail-container)
Azure Event Grid (image-event-subscription)
Azure Function (labelThumbnail)
Azure Cognitive Computer Vision
Azure Search

Azure Search

- Fully managed Azure service for maintaining search indexes
- Offers REST APIs for updating/accessing the indexes
- Replication/Partitioning
  - Disable replication and partitioning for this project
- Upload thumbnail ids and tags

Azure Cognitive Computer Vision

- Image recognition service



- Pricing!!
  - $1.00 per 1,000 images
  - Keep budget for testing
  - Be careful not to exhaust your budget

Azure Monitoring and Alerts

- To mitigate overspending on Azure Computer Vision, we have provided you with an ARM template for Azure Alerts that **you must apply.**
- This template will create an alert that sends an email if you make more than 1000 Cognitive CV calls within 1 min.

Project 2.3 - Hints

- Student VM must be authenticated with Azure, GCP and AWS in order to submit Task 1.
- Review the suggested libraries
- ffmpeg + ffmpy works well in Python
- Test your functions and triggers manually by uploading video files to blob storage
- Review the Azure App Insights and Monitoring Logs for errors
- Ensure that the output file name adheres to the specified format i.e., “video_n.png” and “video.gif”
- If the submitter VM is not authenticated with Azure and is unable to delete, put, and get files from your blob container, or is not able to invoke your Azure functions, you will not be graded
- Make sure that you are using the tags visual feature provided by Azure Cognitive CV and no other feature for generating the labels
- Use temporary folders for storing images and video during a function execution
  - But, remember that functions are stateless.

Project 2.3 - Reminders

- Tag your resources with:
  Key: Project, Value: 2.3
- Azure Cognitive Services are very expensive.
- Your subscription will be disabled if you run out of your subscription budget. Please exercise caution to plan the budget.
- Remember to delete the Azure resource group to clean up all the resources in the end.

Project 3 - Storage

MySQL™

mongoDB

APACHE HBASE

neo4j

SQL

Primers for Project 3

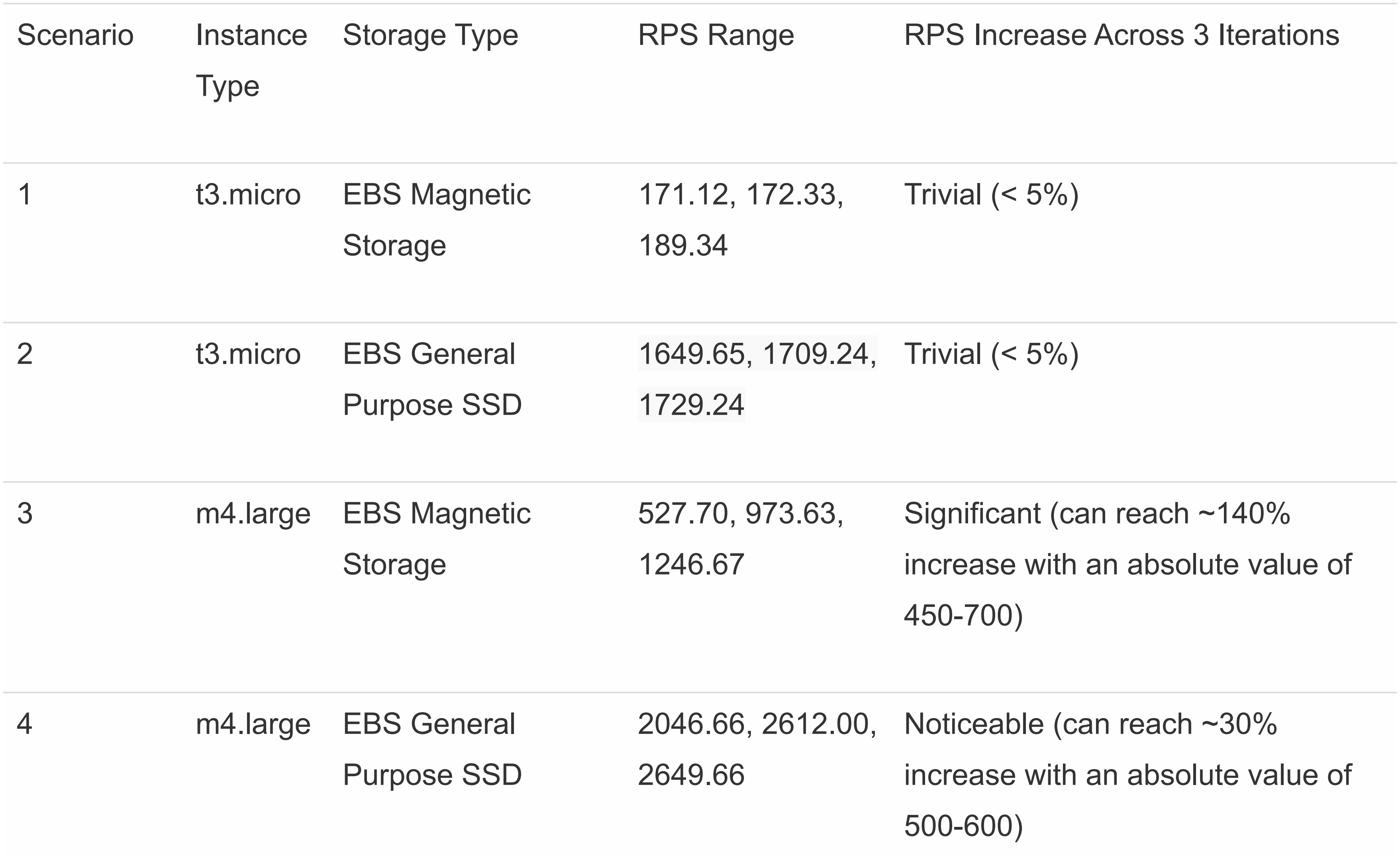
- P3.1: Files, SQL and NoSQL (**Optional**)
  - Primer: Storage & IO Benchmarking
  - Primer: NoSQL and HBase primers
- P3.2: Social network with heterogeneous backend storage
  - Primers: MongoDB
- P3.3: Replication and Consistency models
  - Primer: Intro. to Java Multithreading
  - Primer: Intro. to Consistency Models

Storage & IO Benchmarking Primer

Storage & IO Benchmarking:

- Running sysbench and preparing data
  - Use the prepare option to generate the data.
- Experiments
  - Run sysbench with different storage systems and instance types.
  - Doing this multiple times to reveal different behaviors and results.
- Compare the requests per second.

Performance Benchmarks Sample Report



Scenario	Instance Type	Storage Type	RPS Range	RPS Increase Across 3 Iterations
1	t3.micro	EBS Magnetic Storage	171.12, 172.33, 189.34	Trivial (< 5%)
2	t3.micro	EBS General Purpose SSD	1649.65, 1709.24, 1729.24	Trivial (< 5%)
3	m4.large	EBS Magnetic Storage	527.70, 973.63, 1246.67	Significant (can reach ~140% increase with an absolute value of 450-700)
4	m4.large	EBS General Purpose SSD	2046.66, 2612.00, 2649.66	Noticeable (can reach ~30% increase with an absolute value of 500-600)

What can you conclude from these results?

I/O Benchmarking Conclusions

- SSD has better performance than magnetic disk
- m4.large instance offers higher performance than a t3.micro instance
- The RPS increase across 3 iterations for m4.large is more significant than that for t2.micro:
  - The reason is an instance with more memory can cache more of the previous requests for repeated tests.
  - Caching is a vital performance tuning mechanism when building high performance applications.

Project 3.1 Overview

P3.1: Files, SQL, and NoSQL:

- Task 1: analyze data in flat files
  - Linux tools (e.g. grep, awk)
  - Data libraries (e.g. pandas)
- Task 2: Explore a SQL database (MySQL)
  - load data, run queries, indexing, auditing
  - plain-SQL v/s ORM
- Task 3: Implement a Key-Value Store
  - prototype of Redis using TDD
- Task 4: Explore a NoSQL DB (HBase)
  - load data, run basic queries

The NoSQL and HBase primers are vital for P3.1

Flat Files

- Flat files, plain text or binary
  - comma-separated values (CSV) format: `Carnegie,Cloud Computing,A,2018`
  - tab-separated values (TSV) format: `Carnegie\tCloud Computing\tA\t2018`
  - a custom and verbose format: `Name: Carnegie, Course: Cloud Computing, Section: A, Year: 2018`

Flat Files

- Lightweight, flexible, in favor of small tasks
  - Run it once and throw it away
- Performing complicated analysis on data in files can be inconvenient
- Usually flat files should be fixed or append-only
- Writes to files without breaking data integrity is difficult
- Managing the relations among multiple files is also challenging

Databases

- A collection of organized data
- Database management system (DBMS)
  - Interface between user and data
  - Store/manage/analyze data
- Relational databases
  - Based on the relational model (schema)
  - MySQL, PostgreSQL
- NoSQL Databases
  - Unstructured/semi-structured
  - Redis, HBase, MongoDB, Neo4J

Databases

- Advantages
    - Logical and physical data independence
    - Concurrency control and transaction support
    - Query the data easily (e.g., SQL)
    - ...

- Disadvantages
    - Cost (computational resources, fixed schema)
    - Maintenance and management
    - Complex and time-consuming to design schema
    - ...

Files vs. Databases

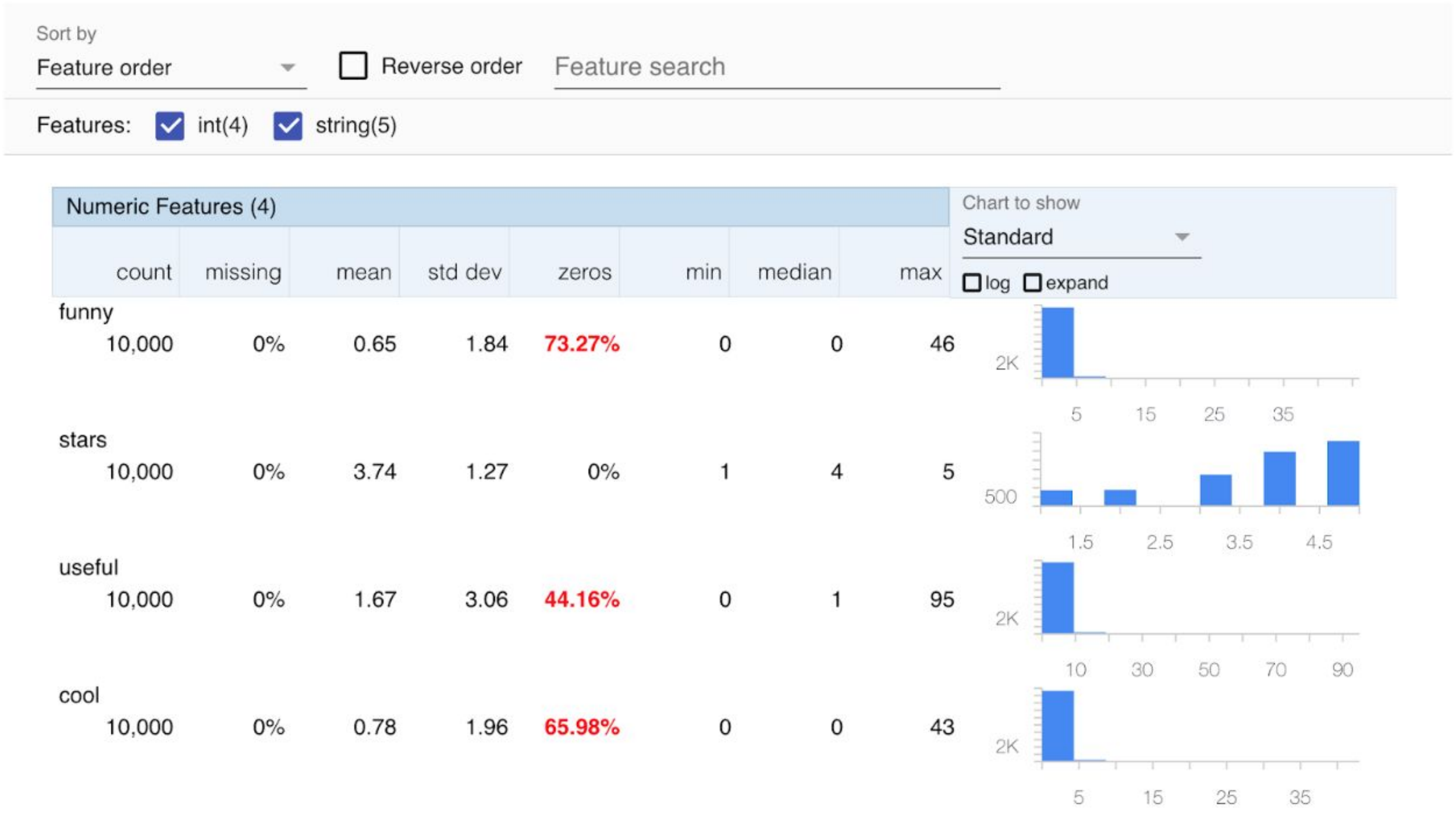
- Compare flat files to databases

- Think about:
  - What are the advantages and disadvantages of using flat files or databases?
  - In what situations would you use a flat file or a database?
  - How to design your own database? How to load, index and query data in a database?

Dataset

- Analyze Yelp's Academic Dataset
    - https://www.yelp.com/dataset_challenge
    - business
    - checkin
    - review
    - tip
    - user

Inspect and visualize data using Facets



Sort by: Feature order

Reverse order

Feature search

Features: ✓ int(4) ✓ string(5)

Numeric Features (4)

Chart to show: Standard

log expand

	count	missing	mean	std dev	zeros	min	median	max
funny	10,000	0%	0.65	1.84	73.27%	0	0	46
stars	10,000	0%	3.74	1.27	0%	1	4	5
useful	10,000	0%	1.67	3.06	44.16%	0	1	95
cool	10,000	0%	0.78	1.96	65.98%	0	0	43

Task 1: Flat File

- Answer questions in runner.sh
  - Use tools such as awk and pandas
  - Similar to what you did in Project 1
- Merge TSV files by joining on a common field
- Identify the disadvantages of flat files
- You may use Jupyter Notebook to help you solve the questions in Python.

Task 2: MySQL

- Prepare tables
  - A script to create the table and load the data is already provided
  - Note: loading the data takes serious time
- Use 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

MySQL Indexing

- **Schema**
  - The structure of the tables and the relations between tables
  - Based on the structure of the data and the application requirement
- **Index**:
  - An index is simply a pointer to data in a table. It’s 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**
- You can build effective indexes only if you are aware of the queries you need
- We have an insightful section about the practice of indexing, read it carefully! **Very helpful for the team project**

EXPLAIN statements 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 statements.

- An EXPLAIN statement on a query will predict:
    - The number of rows to scan
    - Whether it makes use of indexes or not
    - etc.

Object Relational Mapping (ORM)

ORM is a technique to abstract away the work for you to:

1. Map the domain class with the database table
2. Map each field of the domain class with a column of the table
3. 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 adopting ORM

- Separation of concerns
  - ORM decouples the CRUD operations and the business logic code.
- Productivity
  - You don’t need to keep switching between your OOP language such as Java/Python, etc. and SQL.
- Flexibility to meet evolving business requirements
  - ORM cannot eliminate the schema update problem, but it may ease the difficulty, especially when used together with data migration tools.
- Persistence transparency
  - Any changes to a persistent object will be automatically propagated to the database without explicit SQL queries.
- Vendor independence
  - ORM can abstract the application away 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 backward compatible without modifying any business logic code.

NoSQL

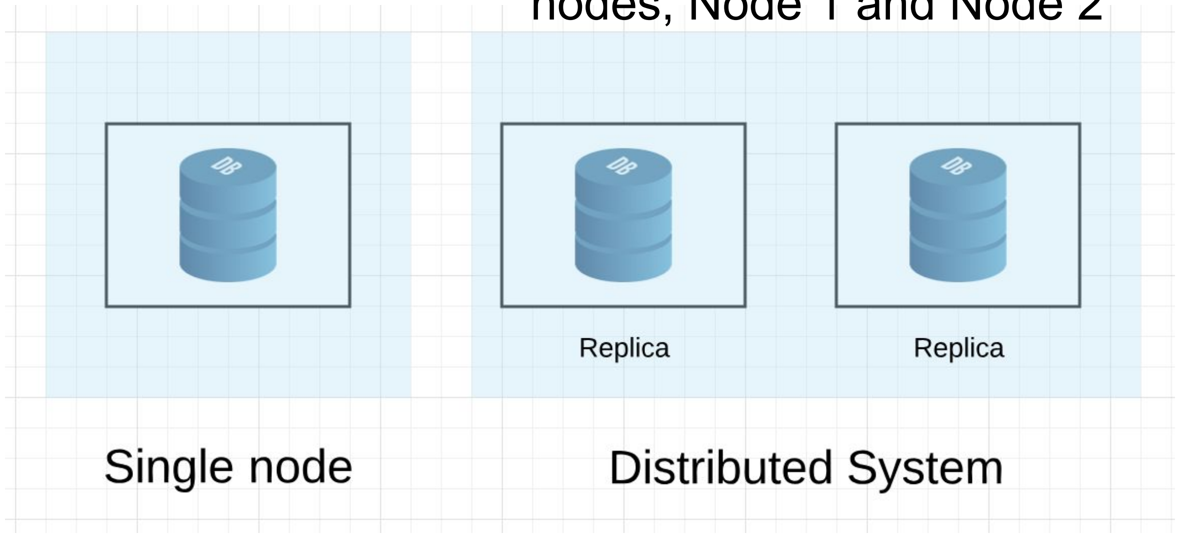
- Non-SQL or NotOnly-SQL
  - Non-relational
- Why NoSQL if we already have SQL solutions?
  - Flexible data model (schemaless, can change)
  - Designed to be distributed (scale horizontally)
  - Certain applications require improved performance at the cost of reduced data consistency (data staleness)
- 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

- The 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

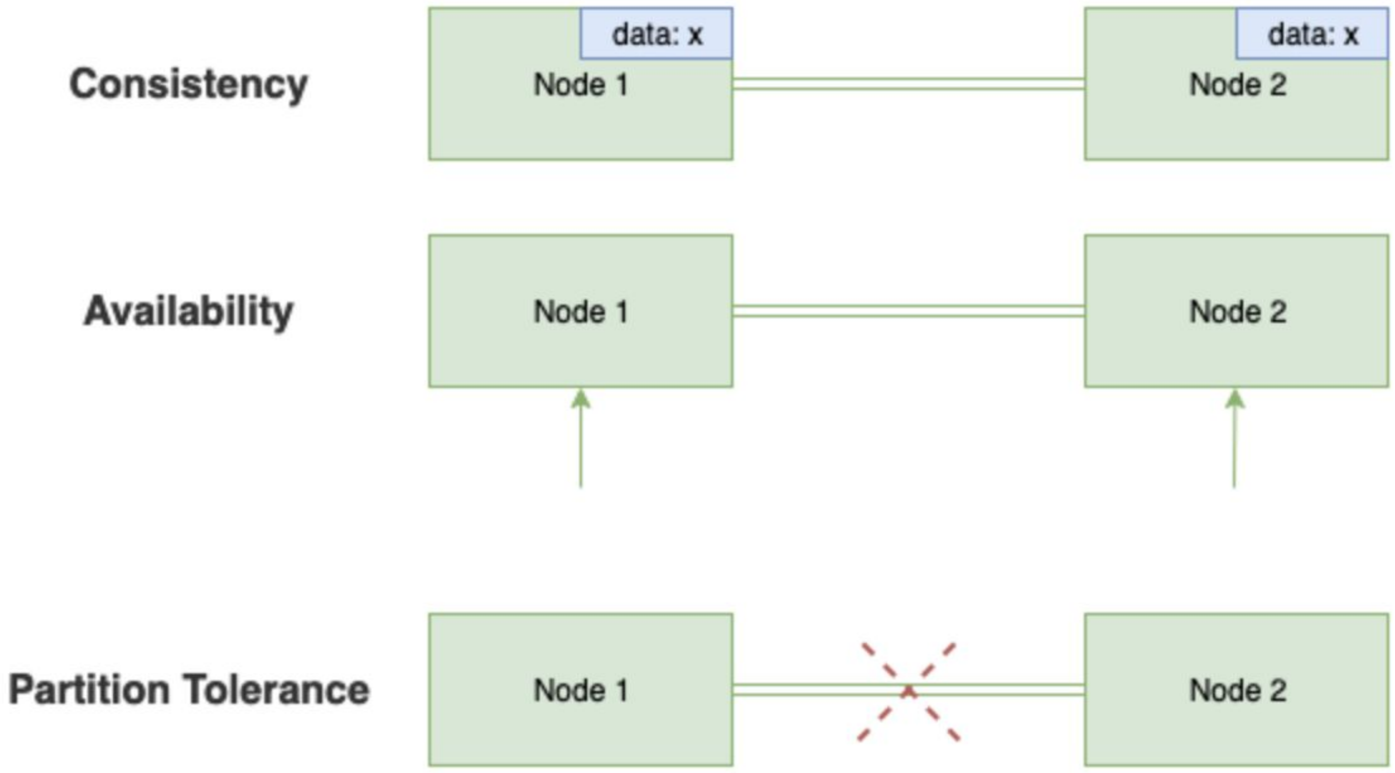
A database, replicated on two nodes, Node 1 and Node 2



Three issues emerge:

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

C, A, P in Distributed Databases

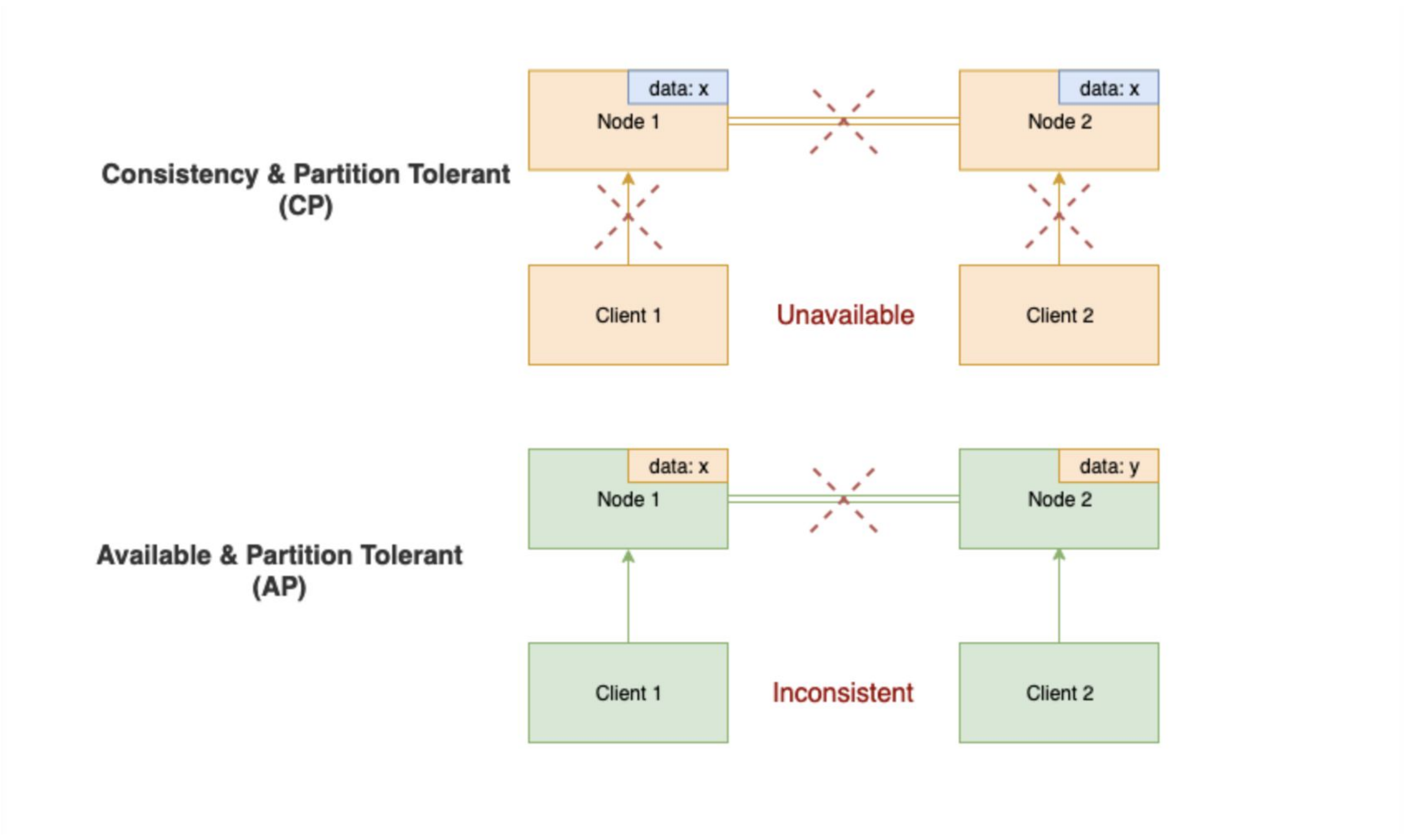


CAP Theorem

Only two out of the three are feasible:

- CA: non-distributed (MySQL, PostgreSQL)
  - Traditional databases like MySQL and PostgresQL have only one server. They meet the requirement of CA and 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 out of three in CAP are feasible



Task 3: Implement Redis

- Key-value store is a type of NoSQL database e.g., Redis and Memcached
- Widely used as an in-memory cache

<u>Your task:</u>

- Implement a simplified version of Redis
- We provide starter code `Redis.java`, you will implement two of the commonly used data structures supported by Redis:
  - hashes and lists
- TDD with 100% code coverage

Task 4: Explore HBase

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

Steps to complete:

1. Launch an HDInsight cluster with HBase installed.
2. Follow the write-up to download and load the data into HBase.
3. Try different querying commands in the HBase shell.
4. Complete HBaseTasks.java using HBase Java APIs.

P3.1 Reminders

- Tag your resources with:
    - Key: Project, Value: 3.1
- An HDInsight cluster is very expensive.
- Your subscription will be disabled if you run out of your subscription budget. Please exercise caution to plan the budget.
- Remember to delete the Azure resource group to clean up all the resources in the end.

Upcoming Deadlines

- **Quiz 5 (OLI Modules 10, 11 & 12)**
  - Due on **Friday**, Oct 4th, 2019, 11:59PM ET
- **Team Project - Team Formation**
  - Due on **Friday**, Oct 4th, 2019, 11:59PM ET
- **OPE - Schedule**
  - Due on **Saturday**, Oct 5th, 2019, 11:59PM ET
- **Project 2.3**
  - Due on **Sunday**, Oct 6th, 2019, 11:59PM ET
- **Project 3.1**
  - Due on **Sunday**, Oct 13th, 2019, 11:59PM ET

Team Formation - Deadlines

Follow the instructions on Piazza carefully

- **By Friday 10/04 at 11:59 PM ET**
  - Identify your team members
  - One team member should form a team on TPZ and all other team members should accept the invitation
    - Completing this step will freeze your team

- **By Saturday 10/05 at 11:59 PM ET**
  - Create a new AWS account ⇒ only used for the team project
  - Update the team profile in TPZ with the new AWS ID `aws-id`

- **By Sunday 10/06 at 11:59 PM ET**
  - Finish reading the Profiling a Cloud Service primer to get yourself prepared for the team project

Q&A