15-319 / 15-619
Cloud Computing

Recitation 6

October 3rd, 2017

Overview

- **Announcements**
- **Last week's reflection**
- Project 2.2, OLI unit 3 module 7, 8 and 9
- **This week's schedule**
  - Unit 3 - Modules 10, 11 and 12
  - Quiz 5 - Friday, October 6th
  - Project 3.1 - Sunday, October 8th

Announcements

- Use spot instances as much as possible
  - e.g. P3.1 HBase cluster
- Be careful about tagging, esp. spot instances
- Protect your credentials
  - Crawlers are looking for AWS credentials on public git repos!
- Start early!

Last Week's Reflection

- OLI: Conceptual Content
  - Unit 3 - Modules 7, 8 and 9:
    - Introduction and Motivation
    - Virtualization (Definition and Types)
    - Resource Virtualization - CPU
  - Quiz 4 completed
- P2.2: Containers: Docker + Kubernetes

Project 2.2
Containers: Docker + Kubernetes

- Build a multi-cloud service to compile and run user code submitted through the front end.
- Project Tasks:
  - Task 1: Docker image.
  - Task 2: Deploy a Kubernetes cluster on GCP
  - Task 3: Container Orchestration and Multi-Cluster Management on GCP and Azure
  - Task 4: Autoscaling and Fault-Tolerance in Kubernetes
- Scalability is one of the key concepts on the cloud

This Week: OLI Content

- 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

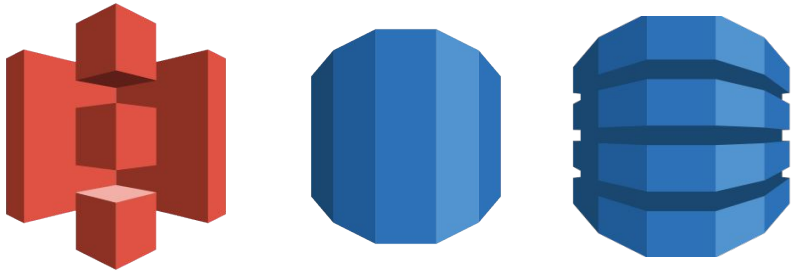
- Understand two-level page mappings from virtual memory to real pages, from real pages to physical memory
- Learn how memory is overcommitted and reclaimed using ballooning
- Study how I/O requests are intercepted at different interfaces
- Map these concepts into your practical exploration with AWS

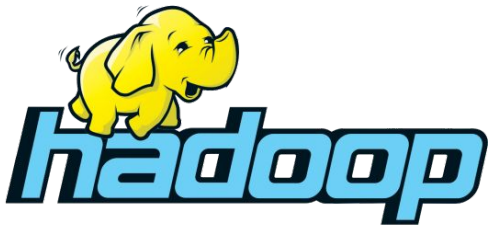
This Week's Project

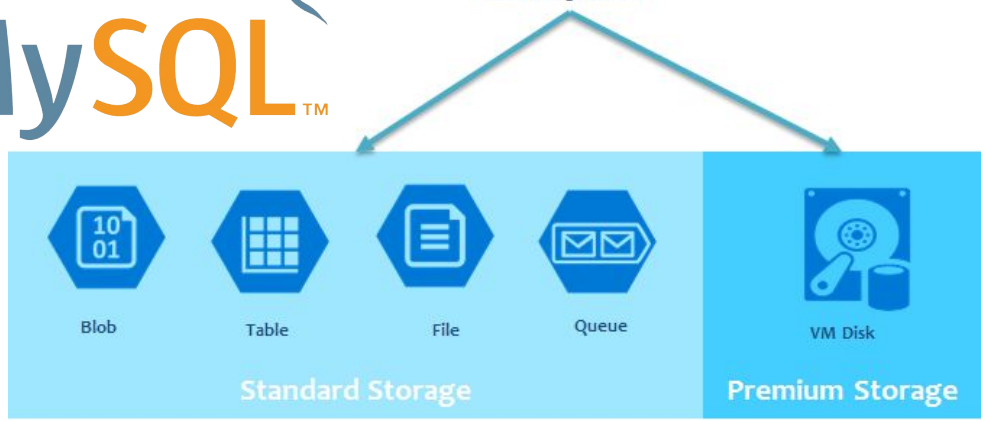
Project 3: Storage and DBs on the cloud

- P3.1: Files and Databases
  - Comparison and Usage of Flat files, MySQL and HBase
- P3.2: Social Networking Timeline with Heterogeneous Backends
  - Social Networking Timeline with Heterogeneous Backends (MySQL, HBase, MongoDB, S3)
- P3.3: Replication and Consistency
  - Multi-threaded Programming and Consistency

Project 3 - Storage







Project 3 Weekly Modules

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

Storage & IO Benchmarking

Link to Storage Benchmarking Primer

- 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	t1.micro	EBS Magnetic Storage	100, 100.5, 100	Trivial (<5%)
2	t1.micro	EBS General Purpose SSD	617.98, 643.99, 634.33	Trivial (<5%)
3	m3.large	EBS Magnetic Storage	309.88, 383.47, 460.89	Significant (can reach 50% with absolute increase of 150-200)
4	m3.large	EBS General Purpose SSD	1367.32, 1653.04, 1722.52	Noticeable (can reach 25% with absolute increase of 300-400)

What can you conclude from these results?

I/O Benchmarking Conclusion

- SSD has better performance than magnetic disk
- m3.large instance has better performance that t1.micro instances
- The RPS increase across 3 iterations for m3.large is more significant than that for t1.micro:
  - The reason is an instance with more memory can cache more of the previous requests for repeated tests.
  - **Caching is also a vital performance tuning mechanism when building high performance applications.**

Project 3.1 Overview

P3.1: Files, SQL, and NoSQL:

- Use Linux tools (e.g. grep, awk) and data libraries (e.g. pandas) to analyze data from given datasets in flat files
- Use relational databases (MySQL)
  - load data, run basic queries
- Use a NoSQL database (HBase)
  - load data, run basic queries

The NoSQL/HBase Primers are vital to P3.1

Flat Files

- Flat files, plain text or binary
  - comma-separated values (CSV) format:
    `Carnegie,Cloud Computing,A,2017`
  - tab-separated values (TSV) format:
    `Carnegie\tCloud Computing\tA\t2017`
  - a custom and verbose format: `Name: Carnegie, Course: Cloud Computing, Section: A, Year: 2017`

Flat Files

- Lightweight, Flexible, in favor of small tasks
  - Run it once and throw it away
- Performing complicate analysis on data in files can be inconvenient
- Usually flat files should be fixed or append-only.
- Writes 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
- NoSQL Databases
  - Unstructured/semi-structured
  - DynamoDB, HBase, Mongo, Google BigTable

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?

Flat File Tasks

- Analyze Yelp's Academic Dataset
  - `https://www.yelp.com/dataset_challenge`
- Answer questions in runner.sh
  - Use tools such as awk, grep, pandas
  - Similar to what you did in Project 1.1, 1.2
- Merge TSV files by joining on a common field
- Identify the disadvantages of flat files

MySQL Tasks

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

MySQL Indexing

- Schema design is based on the structure of the data; index design is 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 them carefully!

EXPLAIN statements in MySQL

- How do you 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.

NoSQL

- Non-SQL(Non-relational) or NotOnly-SQL
- Why NoSQL if we already have SQL solutions?
- Traditional SQL Databases do not align with the need of distributed systems.
- CAP theorem is a conditional trade-off when network partition happens in a distributed system
  - Consistency: no stale data
  - Availability: no downtime
  - Partition Tolerance: network failure tolerance in a distributed system
- Flexible Data Model

HBase (NoSQL) Tasks

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

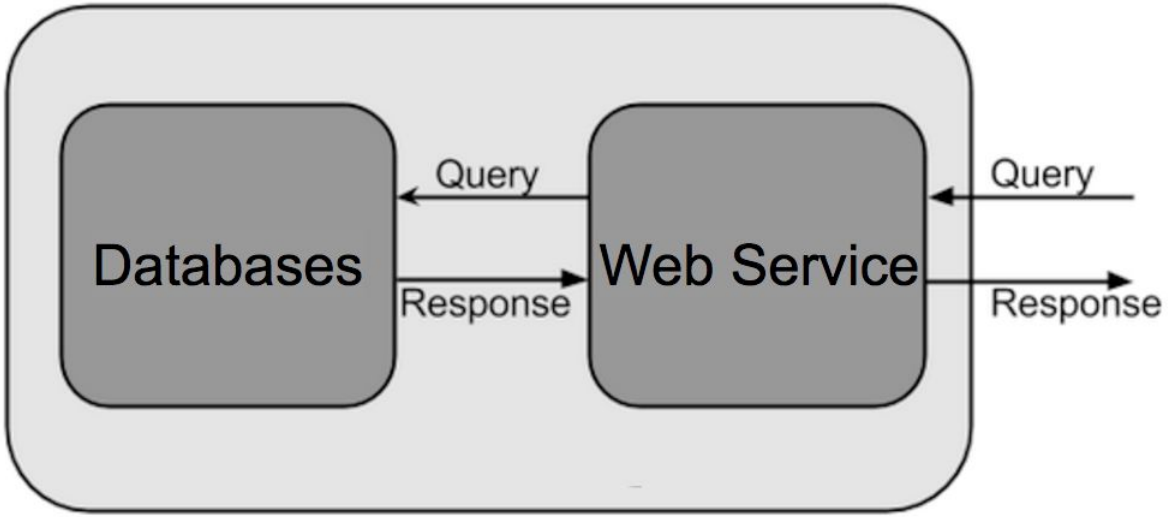
P3.1 Reminders

- Tag your resources with:
  - Key: Project, Value: 3.1

- Stop the submitter instance to take a break and work on it later. Terminate the EMR cluster.

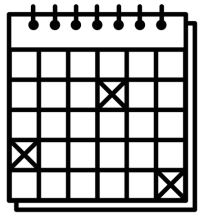
- Make sure to terminate the instance after finishing all questions and submitting your answers.

Team Project Architecture



- We will provide team member suggestions soon.
- Writeup and Queries will be released on Monday, **Oct 9th, 2017.**
- We can have more discussions in subsequent recitations.
- For now, ensure 3-person teams you decide have experience with data processing, web frameworks, database, storage principles and infra setup/hacking. Web UI skill is not needed.

Team Project Time Table



Phase (and query due)	Start	Deadline	Code and Report Due
Phase 1 ● Q1, Q2	Monday 10/09/2017 00:00:00 ET	**Q1: Sunday 10/22/2017 23:59:59 ET** Q2: Sunday 10/29/2017 23:59:59 ET	Tuesday 10/31/2017 23:59:59 ET
Phase 2 ● Q1, Q2,Q3	Monday 10/30/2017 00:00:00 ET	Sunday 11/12/2017 15:59:59 ET	
Phase 2 Live Test (Hbase **AND** MySQL) ● Q1, Q2, Q3	Sunday 11/12/2017 18:00:00 ET	Sunday 11/12/2017 23:59:59 ET	Tuesday 11/14/2017 23:59:59 ET
Phase 3 ● Q1, Q2, Q3, Q4	Monday 11/13/2017 00:00:00 ET	Sunday 12/03/2017 15:59:59 ET	
Phase 3 Live Test (Hbase **OR** MySQL) ● Q1, Q2, Q3, Q4	Sunday 12/03/2017 18:00:00 ET	Sunday 12/03/2017 23:59:59 ET	Tuesday 12/05/2017 23:59:59 ET

Upcoming Deadlines

- Quiz 5: Unit 3 - Virtualizing IO, Memory; Cases

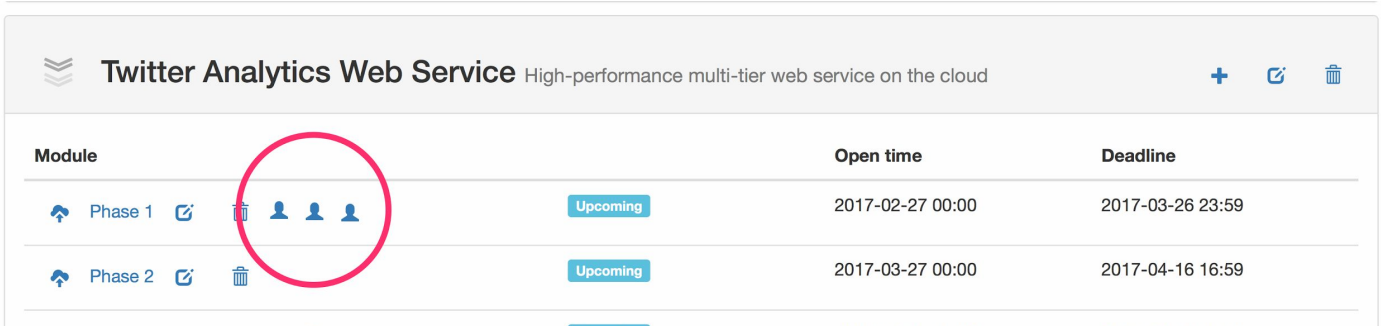
  Due: Oct 6 2017 11:59PM Pittsburgh

- Project 3.1: Files and Databases

  Due: Oct 8 2017 11:59PM Pittsburgh

- Team Project - Team Formation on theproject.zone

  Due: Oct 7 2017 11:59PM Pittsburgh (@Piazza)



Thank you for coming and watching!