ECE1779 Project 1

Introduction to Cloud Computing

▶ Table of Contents

Getting Started

Dependencies

The assignment project requires the following libraries `Flask, gunicorn, requests, mysql.connector, matplotlib.figure`

Installation

`git clone https://github.com/mrchenliang/ECE1779_Project1.git` `pip3 install -r requirements.txt`

Source Database

`mysql -u admin -p ece1779` `mysql> source database/memcache.sql`

Setup and Run

To Start

`sh start.sh`

To Stop

`sh shutdown.sh`

Project Architecture

This assignment project has 2 independent flask instances

- 1 instance is for the backend service running on port 5000
- 1 instance is for the memcache service running on port 5001

The backend service returns web pages and responds to api requests. The memcache service is exposed to the public but is used as an internal service that the backend service calls to configure, add, update and reset the memcache. The memcache also updates the database periodically with the memcache data. The memcache stores the key and the image in base64 and can supports both Least Recently Used and Random Replacement methods.

Web Pages (Port 5000)

- `/` directs to home page
- `/image` directs to get image page
- `/upload_image` directs to put image page
- `/keys_list` directs to get keys list page

- `/cache_properties` directs to get cache properties page
- `/cache_stats` directs to get cache stats page

Backend API Endpoints (Port 5000)

- `/api/upload` post request to upload key and respective file image
- `/api/list_keys` post request to retrieve a list of keys
- `/api/key/<key_value>` post request to retrieve the fiile image of a respective key

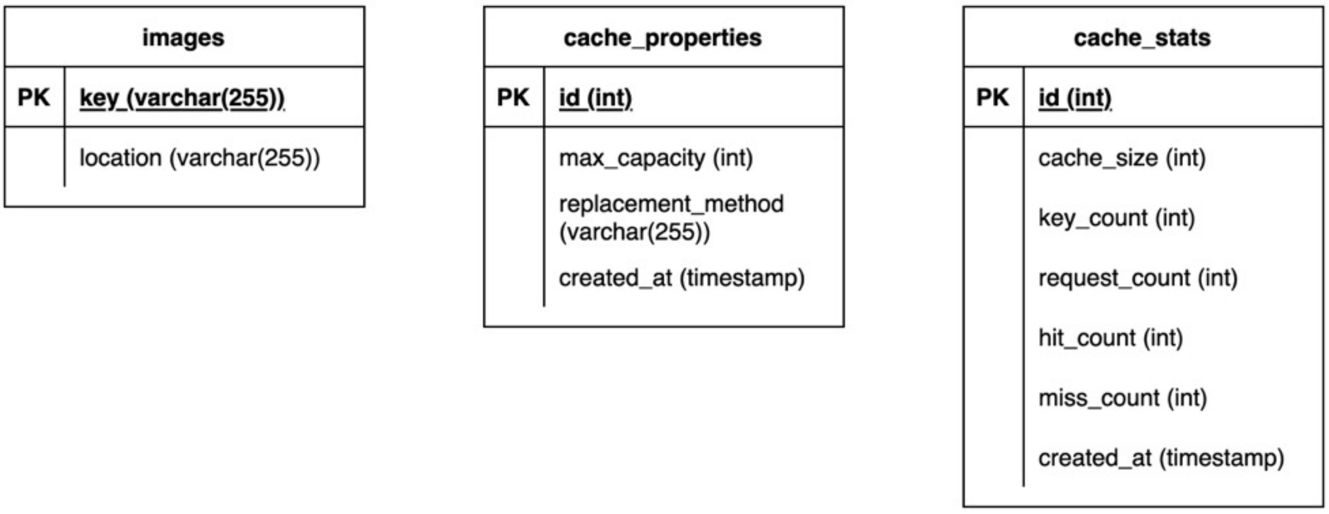
Memcache API Endpoints (Port 5001)

- `/clear_cache` clear memcache and items
- `/refresh_configuration` refresh memcache configuration
- `/put_into_memcache` put key and image into memcache
- `/get_from_memcache` get key and image from memcache
- `/invalidate_specific_key` delete image and respective key from memcache

Database

This assignment project uses a local mysql database to store the following database. There are 3 different tables: 1 for the images, 1 for the cache properties, and 1 for cache stats.

Database Schema



images	
PK	key (varchar(255))
	location (varchar(255))

cache_properties	
PK	id (int)
	max_capacity (int)
	replacement_method (varchar(255))
	created_at (timestamp)

cache_stats	
PK	id (int)
	cache_size (int)
	key_count (int)
	request_count (int)
	hit_count (int)
	miss_count (int)
	created_at (timestamp)

Design Decisions

Independent Flask Instance for Memcache and Backend

- Decision: The decision was to create 2 independent flask instances, one for the memcache, and one for the backend running on port 5001 and 5000 respectively.
- Alternative: The alternative is to have 1 flask instance and have the memcache to run within the backend service, the downside of this is that it is a monolith architecture which is difficult with scaling as there are more opportunities to conflict and overwrite the services.
- Reason: The reason why there are 2 independent flask instances is because they can be seen as individual services that a backend could potentially refactor out to be a microservice environment. In

the future development, the 2 flask instances can be developed independently without interferring with each other and the 2 flask instances will communicate using HTTP requests.

Synchronous vs Asynchronous Operations

- Decision:
- Alternative:
- Reason:

Graphs

Graph 1 20:80 Read/Write Ratio Latency Graph

Graph 2 20:80 Read/Write Ratio Throughput Graph

Graph 3 50:50 Read/Write Ratio Latency Graph

Graph 4 50:50 Read/Write Ratio Throughput Graph

Graph 5 80:20 Read/Write Ratio Latency Graph

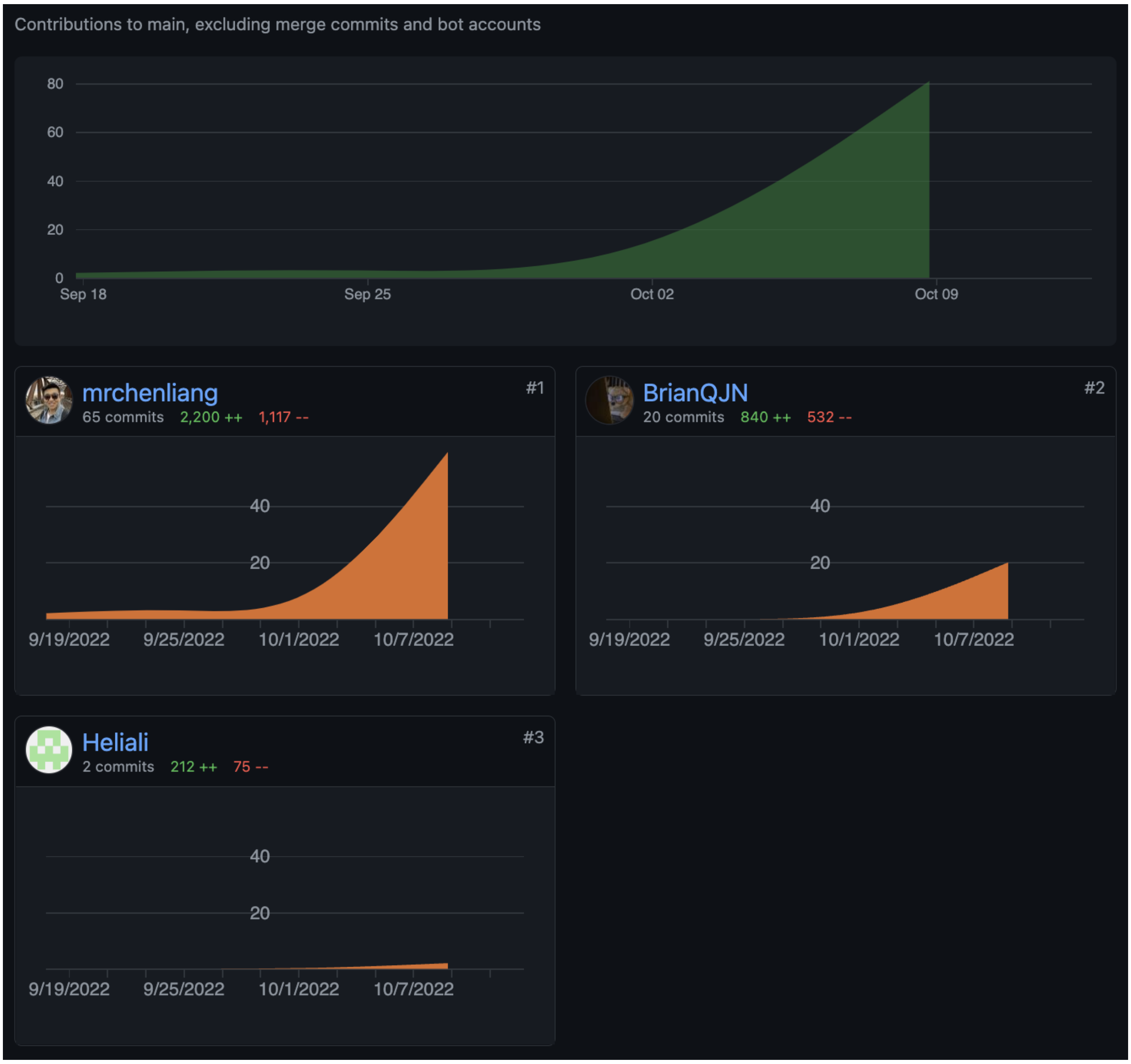
Graph 6 80:20 Read/Write Ratio Throughput Graph

Discussions

Group Members and Contributions:

- mrchenliang
- BrianQJN
- Heliali

Contributions Graph



Contributions to main, excluding merge commits and bot accounts

mrchenliang #1
65 commits 2,200 ++ 1,117 --

BrianQJN #2
20 commits 840 ++ 532 --

Heliali #3
2 commits 212 ++ 75 --