CMPE273: Enterprise Distributed Systems

Lab 2 Kafka And MongoDB

Due: November 5, 2017

This lab covers designing and implementing distributed service oriented application using Kafka. This lab is graded based on 20 points and is an individual effort (no teamwork allowed)

Prerequisites:

- You should be able to run Kafka sample example.
- You should have prior knowledge of JavaScript, default Sessions

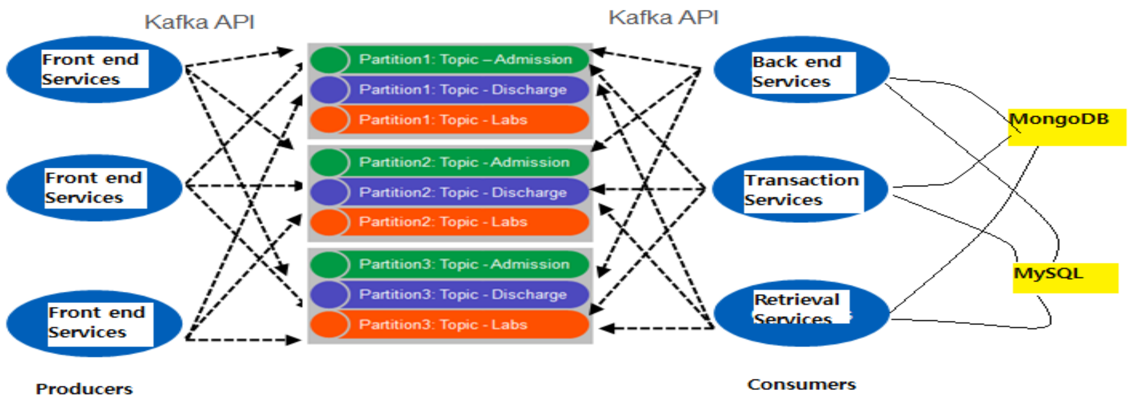
Grading

Late assignments will be accepted, but will be subject to a penalty of -5 points per day late:

- Submissions received at or before the class on the due date can receive maximum

The Assignment

- You will be developing a client and server
- Separate Node into two parts connected via message queues.
  Design "backend services" as consumer and "frontend services" as producer as shown below diagram.



- Implement your own **connection pooling** as discussed in the lecture.
- Use MongoDB as the database. Sessions should be stored in **MongoDB**.
- Passwords need to be encrypted (use **passport**).
- Client and Server should communicate via **Kafka** Streams.
- On, or before the due date, you have to turn in the following:
  - Code listing of client and server
  - Document with architecture of the Kafka interaction in your client/server application, system design description and screenshots

"Dropbox application" to demonstrate RESTful Services (10 pts)

The next node.js based server you need to develop is the "Prototype of Dropbox application". Everyone should create the account on Dropbox and see how it functions. This server should perform the following tasks:

- Basic **Users** functionalities:
    - Sign up new user (at least first name, last name, Email and password)
    - Sign in existing user (Encrypt Passwords)
    - Sign out. Sign Up should have first name, last name, Email and password.
    - Upload a file (small files are good enough)
    - List a file
    - Create a directory
    - Star a folder/directory.
    - Share a folder/directory by email/name/link.

  In order to use the system, a user must sign in first to the system.
- **Users account** should provide basic details such as:
    - **About**: User overview, Work and education, contact info and life events.
    - **Interests** like music, shows and sports.
- Provide **file list and activity report** functionality.
- Provide **Groups (share)** functionalities:
    - Create group
    - Add member in a group
    - Show members in group
    - Assign access permission to a directory
    - Delete member from a group
    - Delete group
- Should perform **connection pooling** for database access (see details below on JMeter section).

The Service should take care of exception that means validation is extremely important for this server. **Good exception handling and prototype similar to actual Dropbox application would attract good marks.**

Client 2 – "Dropbox Client" (2 Points)
Client must include all the functionalities implemented by the web services.
Develop the Client using HTML5 and ReactJS. Client similar to Dropbox will attract good marks.

Testing of the server should be done using JMeter and Mocha. Mocha is a node.js testing framework.

Following tasks to be tested using JMeter: **(2 Points)**
Test the server for **100, 200, 300, 400 and 500 concurrent users (a)** without connection pooling **(b)** with your won implementation of connection pooling and **(c)** DB provided connection pooling and. **Draw the graph with the average time , your analysis of the graph on why, why not and how in the report.**

Following tasks to be tested using Mocha: **(1 Point)**
Implement 10 front-end and 10 back-end test cases for mocha (different test cases from Lab1). **Display the output in the report.**

Create private repository on the GitHub or bitbucket to manage source code for the project. Add description for every commit. Description should consist of one-line overview on what is committed. Include GitHub/bitbucket project link with access credentials in yourreport.

Questions (5 Points)

1. Compare passport authentication process with the authentication process used in Lab1.
2. Compare performance with and without Kafka. Explain in detail the reason for difference in performance.
3. If given an option to implement MySQL and MongoDB both in your application, specify which data of the applications will you store in MongoDB and MySQL respectively

Deliverables Required:

- Submit **architecture diagram of your distributed services**
- Submissions shall include **source code only** for each client/server pair
- Project directory must include the group ID/Name (e.g., Lab2-caffeine)
- Archive the project, and report into one archive file (e.g., zip)
- Do not submit binaries, .class files, or supporting libraries (e.g., junit.jar, javaee.jar) (including them would be **3points** deduction).
- Include the Readme file to document the steps to run the application.
- All the dependencies should be added into package.json file.
- Project report
    - Introduction: state your goals, purpose of system,
    - System Design: Describe your chosen system design
    - Results: Screen image captures of each client/server pair during and after run.
    - Performance: What was performance? Analyze results and explain why are you getting those results.
    - The answers to the

  questionsposed For example:

  Smith is submitting a project. You have provided the following files and source directory:

    - smith-lab2-report.doc
    - lab2/ *(do not send class or jar files)*
  - zip –r Lab2-smith.zip Lab2

Submission

- **On-line submission**: shall include all source zipped with your last names (ex. Lab2- Smith.zip) and your report (smith_lab2_report.doc). Submissions shall be made via Canvas.