FAF.PAD16.2 Autumn 2022

Lab 1: Web Proxy

Handed out: September 7, 2022
Due: October 28, 2022

The Services/The Features

Service nodes are the ones that do the work a client is interested in. They receive tasks, process them and send back responses. Processing requests is usually costly so we imply the help of gateways and caches. Each service has a database. For the services, you'll need to implement the following features:

- ■ SQL and NoSQL databases;
- ■ Tasks distributed across multiple requests;
- ■ Status Endpoint;
- ■ Task Timeouts;
- ■ Service Discovery;
- ■ Adapter with a unified interface to build DB calls based on DB type;
- ★ Priority System;
- ★ RPC;
- ★ Unit Testing;
- ★ Concurrent tasks limit.

The Gateway

The gateway is the node that receives and forwards user tasks to the service nodes. Based on some logic, the gateway caches responses and balances the load of service nodes. Finally, it has a service registry and chooses from registered services when load balancing. For the gateway service, you'll need to implement the following features:

- ■ Load Balancing: Round Robin;
- ■ Circuit Breaker: Trip (log) if a call to a service fails;
- ■ Outbound API - REST;
- ★ Load Balancing: Service Load;
- ★ Circuit Breaker: Remove service if threshold is reached.

The Cache

A cache allows your system to temporary store responses given by your services and serve them without bothering the service nodes. Using caches makes your system more responsive. Usually caches use in-memory storage. For the cache service, you'll need to implement the following features:

- ■ Authentication / Authorization;
- ★ Query Language.

Task explanation / recommendations

The Services/The Features:

- SQL and NoSQL databases - Integrate a new database, different from database that already exists. If you use SQL database, integrate noSQL database and vice versa. Databases must store logically different data;
- Tasks distributed across multiple requests - This functionality can be implemented within the existing service/services. It represents the division of a massive or time-consuming activity into several requests that are executed between 2 instances;
- Status Endpoint - This functionality should be implemented within the existing service/services. Implement endpoint that send information about service (service status, service port, some statistics);
- Task Timeouts - Task Timeout is implemented within the existing service/services. When the service is called, it executes the task, if the task is not completed within a certain time interval, the service sends back "408 Request Timeout". It must be possible to set the time individually for each task;
- Service Discovery - It is a separate service with a static port that receives requests from other services when they start working and sends them to the gateway;
- Adapter with a unified interface to build DB calls based on DB type - It can only be implemented if you have at least 2 databases. This functionality can be implemented within the existing service/services. Depending on the database to which the connection must be made, the service calls the SQL or noSQL database through an interface;
- Priority System - This functionality should be implemented within the existing service/services. You must assign a certain degree of priority to each task that is to be run. The tasks will be executed according to the degree of priority;
- RPC - Data transfer between services based on RPC calls;

- Unit Testing - Write unit tests for at least eighty percent of the functions within the service/services;

- Concurrent tasks limit - This functionality should be implemented within the existing service/services. A maximum number of tasks that it can perform simultaneously must be set for the service. If the service is called but already running the maximum number of tasks, it sends back "429 Too Many Requests".

The gateway:

- Load Balancing: Round Robin - This functionality need to be implemented within the existing gateway. Consecutive distribution of tasks to duplicate services;

- Circuit Breaker: Trip (Log) if a call to a service fails - This functionality need to be implemented within the existing gateway. The circuit breaker overlaps the calls sent to the services, if a service returns an error (exception and/or timeout) then the circuit breaker tries to send similar calls to the service. Set the maximum number of errors sent to the service per time period. In case of success, it returns "ok" and the work process continues normally, otherwise, the error and its details are logged and stops sending calls;

- Outbound REST API - Implement REST API for outbound requests;

- Load Balancing: Service Load - This functionality need to be implemented within the existing gateway. Distribution of tasks to the services with the lowest number of active tasks;

- Circuit Breaker: Remove service if threshold is reached - This functionality need to be implemented within the existing gateway. The circuit breaker overlaps the calls sent to the services, if a service returns an error (exception and/or timeout) then the circuit breaker tries to send similar calls to the service. Set the maximum number of errors sent to the service per time period. In case of success, it returns "ok" and the work process continues normally, otherwise, remove service that return an error.

The cache:

- Authentication / Authorization - This functionality need to be implemented within the existing cache service. Each service that calls the cache must have access only to the data that it has inserted;

- Query Language - This functionality need to be implemented within the existing cache service. Create DSL to access data from cache database.

Reporting

To ensure that no surprises happen at the presentation day, besides the final presentation / repo, you are asked to do weekly status reports on your progress. Every lab you will need to provide a status report in which you discuss your *tangible progress* on the lab. Tangible things include, but are not limited to:

- Commits to your code repo;
- Written research notes;
- Diagrams / drawings.

Make sure that all research materials end up in the project's readme file or in a folder called "docs". Don't forget to upload a link to your public repo of the project on Else, in the "Submit Lab" activity.

Working in Groups

Working in groups is allowed but optional. You can work in groups of 1, 2 or 3 students. The bigger the group, the more features you are expected to implement. *The exact workload should be negotiated with your teacher*, before the first checkpoint.

Checkpoints

To ensure a better workflow during the whole "lab development" period, this lab will have to present 4 checkpoints. These checkpoints will roughly follow the waterfall model of software development, but are open to discussion, so do discuss with your teacher if you would want to apply another approach. *You can pass at most one checkpoint per day*, so plan accordingly.

Passing the first checkpoint involves designing your system. You have to decide the technologies you plan to use during development (i.e. languages, protocols, frameworks etc.), create one Github repository per team and draw a general system's architecture diagram for the project you want to develop.

Getting the second checkpoint implies that you have created an MVP that can receive tasks from clients and route them to service nodes via a gateway node, can process tasks on said service nodes, and can cache the responses on a cache node.

Passing the third checkpoint means that you've developed most of the features that you plan to present at the presentation. The person/team will not be accepted to defend this checkpoint if he does not have a git history or if it is not at an adequate level.

Lastly, the fourth checkpoint is represented by the presentation of the laboratory and the entire work process. The presentation will follow a strict schedule. During the presentation, you are required to:

- State the mark you are aiming for;

- State the features you have implemented;
- Demonstrate the implemented functionality;
- Pass the individual code review;
- Discuss final mark.

Each team will have at their disposal 15/22.5/30 minutes to present their work (for teams of 1/2/3 members respectively). If by the end of the specified time slot you didn't finish presenting, the presentation is considered finished and I mark you (or not) based on what I have. Obviously, the individual code review and git history have the most influence on passing / not passing.

To receive a mark for this laboratory work you need to receive a passed mark on all four checkpoints.

Grading

Team size	Mark 10	Mark 5
1	The Service: 3■, 2★ The Gateway: 2■, 1★ The Cache: 1■ or 1★ Personal Service features: 1	(The Service: 1 in total or The Gateway: 1 in total) and (The Cache: 1 in total or Personal Service features: 1)
2	The Service: 4■, 3★ The Gateway: 3 in total The Cache: 1■ or 1★ Personal Service features: 2	(The Service: 2 in total or The Gateway: 2 in total) and (The Cache: all or Personal Service features: 2)
3	The Service: all The Gateway: REST API + all stars The Cache: all Personal Service features: 5	(The Service: 3 in total or The Gateway: 3 in total) and (The Cache: all or Personal Service features: 3)

Readings

[1] Mark Smallcombe. "SQL vs NoSQL: 5 Critical Differences". https://www.xplenty.com/blog/the-sql-vs-nosql-difference.

[2] Phil Sturgeon. "Understanding RPC Vs REST For HTTP APIs". https://www.smashingmagazine.com/2016/09/understanding-rest-and-rpc-for-http-apis/.

[3] Martin Fowler. "CircuitBreaker". https://martinfowler.com/bliki/CircuitBreaker.html.

[4] MDN Web Docs. "HTTP caching". https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching.

[5] Julien Le Coupanec. "Redis Cheatsheet - Basic Commands You Must Know". https://gist.github.com/LeCoupa/1596b8f359ad8812c7271b5322c30946.

Good Luck!