Design Plan Distributed Systems Project

Period II 2022

Veli-Matti Laamala, Edgars Gaisins, Simon Lechner

Selected Topic

We decided to work on a distributed webshop for our project. It is very common for e-commerce applications to be distributed among several nodes to allow scaling of the application. Further, moving the server closer to the client will allow for faster responses and therefore a better consumer experience. Running separate servers for the user and product services is greatly beneficial for the fault tolerance of the system, if one part fails the other one will still run normally.

Description

A simple webshop with its necessary components is set up. These include storage for user based data and basic services on them as well as product related information and their services. These two parts can easily be split onto two distinct servers with their own ip addresses. The actual shopping carts are then a distributed state which connects a user with their corresponding products. This architecture provides a good base for further development, to scale the application simply more servers for each field can be added. Requests are forwarded by a reverse proxy, this single connection node hides the distribution from the client and therefore from its users.

Nodes and Roles

The toy distributed system consists of four nodes, a simple client, a reverse proxy, one server for consumer data/services and finally one server for product data/services.

- Client: A simple client is set up using react, it is mainly used for testing purposes and to showcase the system. It is not part of the minimum requirement of three nodes since it only communicates with the reverse proxy and not directly with the servers. The client offers functions related to user management as well as the actual shopping.
- Reverse Proxy: The reverse proxy is a single node connection point for the system. It forwards requests depending on their topic to the corresponding server. The reverse proxy could also be used as a load balancer if the project is further developed.
- Consumer server: This server stores data related to consumers such as their name and email. Basic services on this data will be provided.
- Product server: Data related to the products of the webshop is stored on this server. Just like the other server a database is used for persistent storage.

Messages

The client sends requests to the reverse proxy which then forwards these to the right server. The servers are based on a REST API which allows for simple resource management. The shopping task requires connecting users and products, this is achieved by communication between the two servers.

General Comments

We have decided to implement naming & node discovery, synchronization & consistency and fault tolerance out of the four options. Consensus between all nodes is not required in a webshop since the third node, the reverse proxy, is only forwarding massages.

Naming and node discovery is part of almost all distributed systems, it is important to connect the nodes when starting the application.

Synchronization and consistency, since the shopping data is split up into user and product data the shopping cart has to be consistent between both servers.

Fault tolerance is enabled to a certain degree with the chosen way of splitting the application. If the user server fails, products are still visible. If the product server fails on the other hand only user operations are possible. Fault of the reverse proxy will lead to a completely unusable application since the client will no longer be able to send requests.

Due to the usage of the reverse proxy the application could be further extended into multiple servers for users as well as for products. The functions of the reverse proxy would be extended into a load balancer which allows for good scalability.

After some online research we have decided to deploy the application using kubernetes since it provides many of the required functionalities.