Final Project

Distributed Systems

CT30A3401

Vili Huusko

0544718

22.04.2022

Contents

1. Introduction

In this project I developed a client-server application, that finds the shortest path between two Wikipedia articles. It does not, however, keep track of the path, nor does it keep track of the depth of the search.

The client is developed using Python 3.8.3 and the backend is done using Node.js version v.16.13.1. Data transferring between the client and the server is done using hypertext transfer protocol (http) and JavaScript object notation (JSON) data format.

The backend has a master process, which listens to oncoming requests. Upon a request, it forks child workers based on the number of threads in the system (server). These child workers get a list of Wikipedia links based on the user input (the first title given) from which the workers search for the endpoint article.

The client has just one process that listens for the user input. After it gets the required inputs, it connects to the server via http.

The system assumes that the Wikipedia API can be reached. Another assumption is that only one client is connected to the server at a time.

2. System Overview

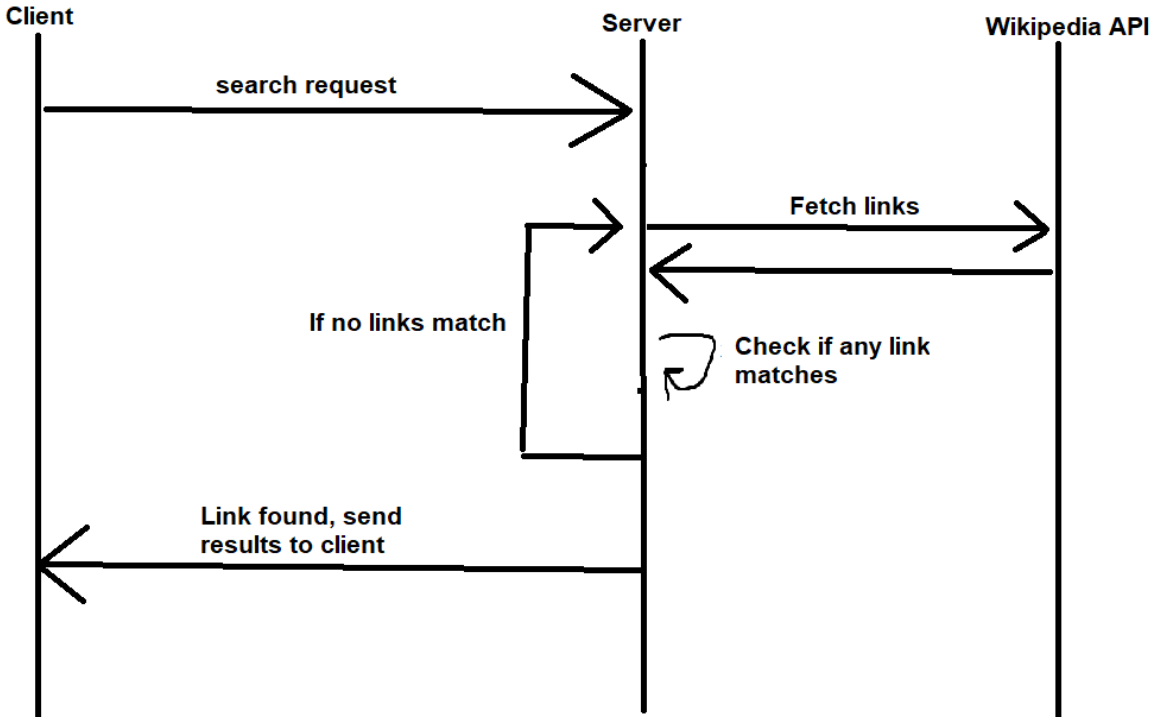

Figure 1: System sequence chart

The sequence diagram shows the basic functionality of the system. First, the client sends an http request to the server, which contains the start and end Wikipedia article titles. After that, the server fetches a list of links from the Wikipedia API and distributes it across several workers.

These workers conduct a breadth-first search from the list of links and fetch more Wikipedia links if the link doesn't match the required parameter.

After the article is found, a worker informs the master process of this, after which the master process terminates all of the child processes and sends the result to the client.

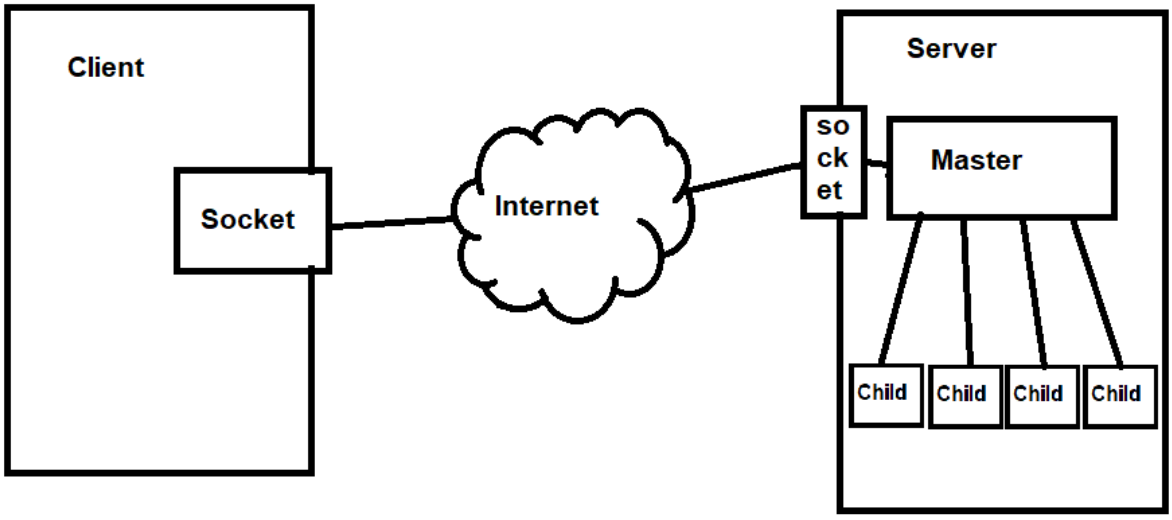

Figure 2: Overview of the architecture of the program

The architecture figure shows the usage of workers to handle the task. The only component absent in the figure is the Wikipedia API server. This is excluded for clarity reasons.

3. Requirements Implemented

Functional requirements:

- Fetches Wikipedia article links from the API.
- Finds out if there's a path between two Wikipedia articles.
- Presents the time that went into the search.
- Presents the link that led to the result.
- Error handling.
- Communication with a client through the internet.

Non-functional requirements:

- Good performance.
- Good user experience through a CLI client.
- Transparency.

4. Development Approach

The system uses the Wikipedia API to fetch the links for the breadth-first search.

The system uses child units to divide the search problem into smaller problems instead of dividing the oncoming requests into different threads. This is done to increase performance with the search.

5. Distributed system properties

The system has no vertical scaling whatsoever. However, some horizontal scalability could be achieved with multiple servers and a balance loader.

The system implements transparency. It does this by having a CLI client that the user interacts with. The distributed nature of the system doesn't show in any way to the end user.

Error handling is done through the usage of try-catch blocks. The client's only networking call that could fail is handled, and the server's multiple network calls (to the client as well as to the API) are handled as well. The system also has a failsafe timer (30 minutes) that stops execution of the search if nothing has been found by then.

6. Notes

I referred the Cluster documentation and the Wikipedia article of breadth-first search when developing the system.