Project report: Stage 1
COMP3100 Distributed Systems, S1, 2023
SID: 46276688, Name: Le Minh Pham

1 Introduction

A distributed system, such as a computer cluster or a (cloud) data center, often deals with various types of jobs (or tasks) that have different resource requirements in terms of processing power (CPU), memory, and disk capacity. After the jobs are submitted to the system, they are scheduled to the servers, the main component of the system, which allocate the necessary resources to complete the jobs. Since servers of a particular distributed system are often heterogeneous in terms of their resource capacity, their performance will change in accordance with the scheduling policy used. Therefore, to optimize a system, we need to configure an appropriate job scheduler.

The aim of this project is to develop a new job scheduler in a simulated distributed system provided by a discrete-event simulator (ds-sim). The project is divided into two stages. In stage 1 of the project, I aim to design and implement my own version of ds-client in ds-sim that works solely as a job scheduler. The scheduling algorithm implemented is Largest-Round-Robin (LRR).

In this report, I discuss the system overview, design, and implementation in Section 2, Section 3, and Section 4 respectively. The repository for this project can be accessed via the following link: `https://github.com/LeMinhPham/COMP-3100-S1-2023-Assignment`

2 System Overview

Client is my implemented version of ds-client in this project. Client follows ds-sim communication protocol and acts as an LRR scheduler, which schedules jobs to the servers of the largest type in a round robin (Figure 1).

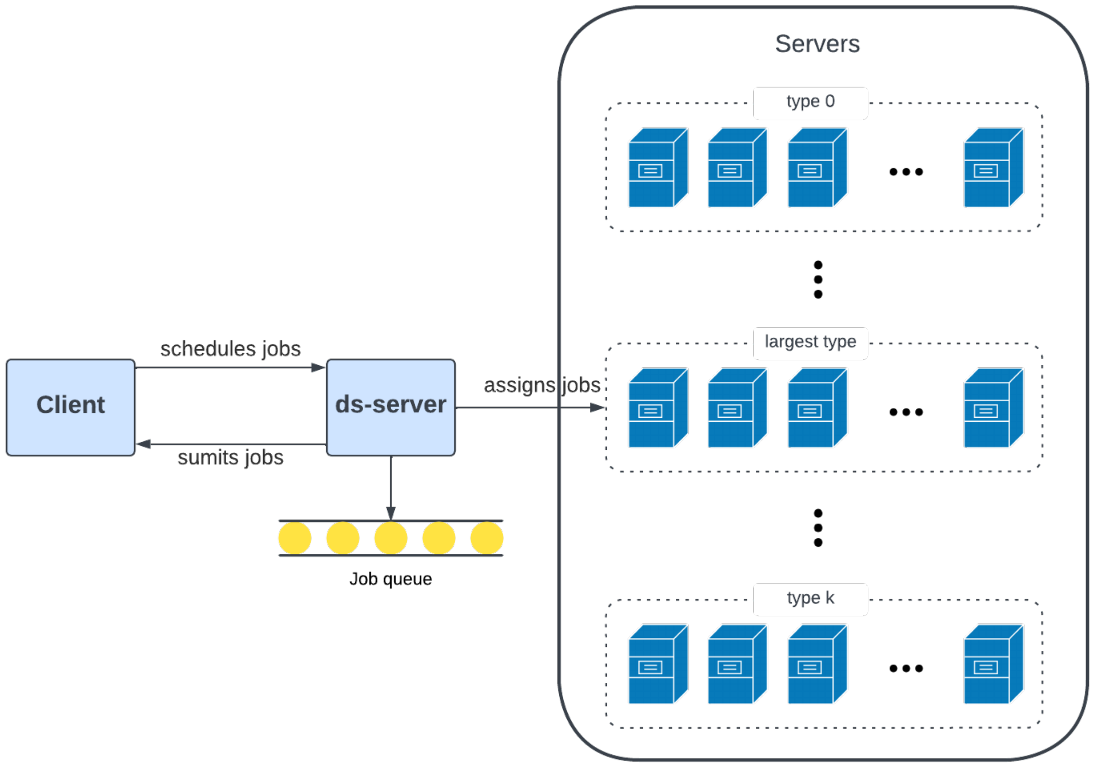

Figure 1: LRR Scheduler in ds-sim

Briefly, in the simulation, ds-server is responsible for managing both job submission and job execution, while Client solely acts as a job scheduler for ds-server. In general, the simulation flow is as follows:

- Initialization: ds-server and Client exchange 'HELO' messages to initialize the simulation.
- Preparation: Client reads system information and determines servers of the largest type.
- Scheduling: Client makes scheduling decisions based on the information read until there is no more job submitted by ds-server.
- Conclusion: ds-server and Client exchange 'QUIT' messages to end the simulation.

3 Design

Since the requirement of stage 1 is simply designing and implementing a vanilla version of an LRR job scheduling client-side simulator, I have not considered any constraints, settings, or policies. I only assume that there will be no server failures during the simulation. In other words, once a job has been scheduled to a server, it is guaranteed that the job will be completed. Therefore, Client can safely disconnect from ds-server once it has appropriately scheduled all jobs.

4 Implementation

Client is implemented in Java and consists of two main components: communication and LRR scheduling algorithm.

4.1 Communication

Client connects to ds-server by using a Socket. The Socket establishes the connection to 127.0.0.1 IP address and port 50000, which is the default IP address and port number of ds-server. After the connection is established, Client communicates with ds-server through a DataOutputStream (for sending messages) and BufferedReader (for receiving messages) and starts scheduling jobs through these channels. The message-exchanging process follows ds-sim simulation protocol.

4.2 LRR

The LRR algorithm is a simple scheduling algorithm that sends jobs to servers of the largest type in round-robin order, i.e. servers are arranged equally and repeatedly from start to end. For example, if the largest server type has five servers indexed from 0 to 4, each server will be scheduled a job in turn from 0 to 4. Once all servers have been scheduled for a job, we cycle back to server 0 and repeat the scheduling process.

server 0 receives job 0

server 1 receives job 1

server 2 receives job 2

server 3 receives job 3

server 4 receives job 4

server 0 receives job 5

...

Conveniently, servers in ds-sim are properly indexed in their respective types. Therefore, to implement the LRR algorithm, we only need to determine the largest server type, which has the largest number of CPU cores, and maintain the server index to schedule a job to.

To configure the largest server type, Client uses the "GETS All" command to obtain the servers' information from ds-server, then chooses the first server with the largest number of CPU cores. To schedule the jobs to the correct server in round-robin order, Client maintains a server index that starts at 0. When a job is submitted, Client sends the job to the server with an index that matches the current server index stored, increments the server index by one, and wraps the server index with the maximum number of servers using the module operation.

At the beginning, Client initializes server index to 0

job 0 arrives, Client sends job 0 to server 0 and increases index to 1

job 1 arrives, Client sends job 1 to server 1 and increases index to 2

job 2 arrives, Client sends job 2 to server 2 and increases index to 3

job 3 arrives, Client sends job 3 to server 3 and increases index to 4

job 4 arrives, Client sends job 4 to server 4, increases index to 5, and wraps index back to 0

job 5 arrives, Client sends job 5 to server 0 and increases index to 1

...