Project report: Stage 1 COMP3100 Distributed Systems, S2, 2022

SID: 45334625, Name: Shakeel Mohammed

1 Introduction

What is this project? This report presented is to describe the overview, design, and implementation of a scheduling system that implements the Largest Round Robin algorithm whilst distributing jobs to servers in a simulated distributed system (provided by MQ University). The results generated by the use of this project are to be compared to a reference implementation (also provided by MQ University).

At it's core, a distributed system is a system that has various components which are spread across a network. This provides many advantages such as efficiency and redundancy, as well as introducing complications such as an increased complexity, synchronisation and replication.

In a distributed system, the load is distributed across multiple machines to achieve fast and more reliable results. Each distributed system requires a component to orchestrate and distribute each request.

This project acts as an orchestrating component of a distributed system.

What is the goal? The goal for this project is to have a client which connects to the server, and makes scheduling decisions based on the desired algorithm.

2 System Overview

The system is comprised of two main components:

1. the ds-sim Server (provided by MQ University) - initiates the request to process each job, as well as simulates a distributed system.
2. the Client (this project) - handles the request to process each job and schedules jobs for processing.

2.1 Communication Protocol

The two components communicate via a Socket, which uses TCP at the network layer. See here cite[https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html]

The protocol is as follows:

1. the ds-sim server is started, creates a simulated system based on the confile file passed in, and waits for incoming socket connections
2. the client is started and initiates a handshake with the server
3. server replies with HELO
4. client sends AUTH command
5. server accepts and replies with HELO

6. the server writes information on the simulated system to a ds-system.xml file
7. in a loop, the following happens:
   (a) the client sends REDY command
   (b) the server responds with either:
      i. a new job
      ii. a completed job
      iii. an error
      iv. a "no jobs available" response
   (c) the client handles as follows:
      i. queries the server to find servers capable of handling the jobs (GETS CAPABLE command) and selects the correct server to handle the job based on the Largest Round Robin algorithm.
      ii. ignored as completed jobs do not require handling
      iii. the logs the error and stops the job scheduling process.
      iv. this response will break the loop
8. once the loop is complete (no more jobs left to process), the client sends QUIT command
9. the server replies with QUIT
10. the connection is closed gracefully

3 Design

The design must cater for the connection between the two main components; the server and the client, as well as break down the current state of the simulated ds-sim system at any one time to handle incoming jobs. The two main entities in ds-sim are Servers and Jobs. These are the entities that are used to make scheduling decisions.

3.1 Servers

In a distributed system, a server is a compute resource equipped with its own CPU, memory, and disk. They can be either hardware/physical or virtual servers. In the case of the ds-sim system, each simulated server is virtual. Each server has it's own serverId, serverType, limit, bootUpTime, hourlyRate, cores, memory, and disk attributes.

3.2 Jobs

In this context, a Job can represent a task, or a single linux process. They are to be seen as rigid. i.e. if a job have compute requirements of 2 Cores, 200MB of memory and 1GB of disk space, and will take 10 seconds to complete, scheduling it on a server with twice the amount of the required resources will not result in a faster completion time. It will still take 20 seconds to complete. Each job has

it's own jobId, type, submitTime, estimatedRunTime, cores, memory, and disk attributes.

Processing information on each of these entities is crucial for our design to be able to make the correct scheduling decisions to achieve our goal: scheduling based on Largest Round Robin.

3.3 Scheduling

Processing information on each of these entities is crucial for our design to be able to make the correct scheduling decisions to achieve our goal to schedule jobs based on Largest Round Robin, which operates by scheduling jobs to servers that are of the type in the system have the highest number of cores, in a round robin manner. i.e. first job to server 1, second job to server 2, and so on.

3.4 Components

The design achieves job scheduling by Largest Round Robin by the use of the following components:

ConfigDataLoader

A singleton object which serves as a configuration parameter store. It reads key/value pairs from the config.properties file and makes those parameters available to our application. It allows the client to be re-configured without requiring re-compilation.

ClientServerConnection

A singleton object which handles the client/server socket connection. It provides an interface for the rest of the application to send and receive messages to the ds-sim server.

Orchestrator

The compomonent responsible for making scheduling decisions. It currently implements the Largest Round Robin algorithm. It can easily be extended to support additional alogrithms. The Orchestrator uses the methods provided by the SimulatedSystem, SimulatedServer, and Job classes to decide which scheduling decisions should be made.

SimulatedSystem

The component responsible for containing information on the simulated system which the ds-sim server provides. It provides information on the system as a swarm of servers. i.e. the largest type of server, list of servers, etc.

SimulatedServer

Each instance of this component represents one server which exists in the ds-sim server. It contains information on the type of server, number of cores, number of jobs running, etc.

Job

A Job object represents one job which has been sent to the client by the ds-sim server. It contains all of the attributes that define a job. Each job object also contains a method which can be used to generate a query that can be sent to the ds-sim server in order to query a list of servers capable of processing the job. A job is instantiated by the use of a JobInformation and a JobInformation component as they allow for a standardization layer to the creation of each Job component. This is required because the ds-sim server is capable to providing a job in multiple different formats.

3.5 Constraints

This project requires a Java 1.8 to be run, which can be downloaded by following the instructions here, [https://docs.datastax.com/en/jdk-install/doc/jdk-install/installOpenJdkDeb.html]

3.6 Considerations

As per the ds-sim protocol, there are two methods by which a connecting client could retrieve information about the simulated system:

1. ds-system.xml file - once a client has connected and a handshake is successful, the ds-sim server will create a ds-system.xml file which contains information on the simulated system in an xml format. This can then be parsed by the client used to make scheduling decisions

2. GETS command - a client is able to query the ds-sim server for a list of servers (GETS AVAIL and GETS CAPABLE). This can be a complete exhaustive list containing all of the simulated servers that can eventually become available, or a more precise list of server capable of handing a job based on cpu, memory and disk, which are immediately available for use.

This design uses the GETS CAPABLE command to increase efficiency and search only for the servers that are capable of handling the particular job ready for processing.

4 Implementation

The design must cater for the connection between the two main components; the server and the client, as well as break down the current state of the simulated ds-sim system at any one time to handle incoming jobs. The two main entities in ds-sim are Servers and Jobs. These are the entities that are used to make scheduling decisions.

5 References