# Internal Virtual Job Scheduling System

## Developers

Joel Duff – 44905866

Ngoc Lan Tran – 44480636

Warren De Silva – 44912838

## Introduction

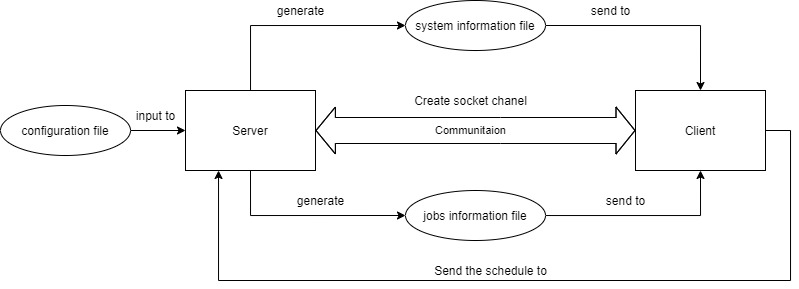
Distributed system in computing is diverse about size and types. It can be a single computer or a large network between multiple computes as data centres in the world. There are two main components to build the distributed system is server and client. Servers are connected from many single systems to implement jobs to clients. However, this project focuses on the client side when designing and developing a working model of client is job schedule.

In a system, there are many jobs from multiple clients at same time; and each job have different requirements from server to solve this job such as memory, CPU cores and disk space. This can lead to stuff at bottle neck when there are several numbers of jobs are sent at the same time. Thus, the schedule is suitable technique to solve this problem.

This project is developing depend on a discrete event simulator (ds-sim) to create the job schedule system which connects to servers, gains jobs and schedules them. This also dispatch the jobs to the largest server which is identified by the number of CPU cores.

## System Overview

The job schedule system includes two sides are server and client and base on ds-sim which is distributed systems simulator. Both sides initialise a communication by handshake and implement a loop to finish jobs which is known as discrete event simulation. After handshake between client and server to authenticate the communication’s information, server creates a list of job and transfers to client with some options i.e., JOBN, JCPL, RESF; after client finishes its job, server will run all jobs depend on the priority in client’s schedule. In client side, client receives the messages and job list from server and analyse them. In schedule policy, client will send all jobs to the largest server at the current running time, it is changing if there are amount of number of jobs are running at one server.



## Design

In the job scheduling systems, there are three main simulator components are jobs, server and client.

Job in this system requires some attributes from servers such as the number of CPU cores, memory in RAM and disk space. Each job has id as sequence number depend on increasing submission time, type which bases on running time value, submission time and estimated running time.

Server also have its value on CPU cores, memory and disk space which will provide to run the jobs; id and type is also identified by submission and running time. Furthermore, server has its own component such as limited number of servers in same type, the booting time by type and the renting cost per hour.

In this system, client is the main simulator designed and developed. The client uses all-to-largest algorithm (largest current server) to schedule the jobs. Upon receiving a job, it polls ds-server for the list of servers and sends the job to the largest active server at the current time. The difference between the largest server overall and the largest server at the current time is that as a server receives more jobs, it will have less resources available, making it functionally smaller. Eventually, when the largest server has enough jobs, it will be considered functionally smaller than a server with less maximum resources, but more available resources.

## Implementation

The implementation of the Internal Virtual Job Scheduling system was conducted through multiple java files, each undertaking specified tasks that would allow the system to function as designed. In total, four java files were created, with one file conducting all the job scheduling tasks, and the remaining three files conducting smaller but necessary tasks for the system. The four files created are as such:

* Connection.java
* Server.java
* ServerCluster.java
* ds\_client\_test.java

The Connection java file, is used to create the connection between the client and the server. The Connection java file’s primary job is to connect the client to the server, and create the ‘handshake’ between both sides, by sending messages to server, and receive and read the messages from the Server. Any messages sent from the client to the server, is sent through this file’s functions. The software libraries that were utilised in this java file are the input/output libraries, the network library and the utility library.

The Server java file, has only one function, and that is to get the attributes of the server. The attributes are then stored in their respective variables utilizing the necessary data types. Some of the types of attributes that are stored are the types of servers. It accepts a list of arguments or whatever the XML parser gives it. Attribute data stored in the variables are then used in other java files, to help assist with identifying the desired server to be used. This file utilised the Document Object Model Element from the W3C standard.

The SeverCluster java file is used to sort the list of servers that are passed through the xml Parser. The list of servers is sorted using the information that was stored in the Sever java file, with the “Core Count” Attribute being used sort the list, from smallest to largest. This was implemented to allow easier job scheduling to the largest server (the largest server being the server with the most amount of core counts).

The ds\_client\_test java file is the main file in which a large majority of the functions, that allow this system to function, is implemented. The libraries that were utilised in this file are the input/output libraries, the network library, the utility library, the Document Object Model library from the W3C standard and the XML library. This file holds all the messages that are sent to the servers, through the Connection.java file, that help establish the connection between the client and the server. There is also a ‘for’ loop and ‘switch’ method that utilises argument parsing to get technical information such as the port being utilised or the system path to the files. A Java exception try and catch method is used to parse the XML file information, which is then sent to the Server.java file to create a list of servers holding each server’s information, which is sent to the ServerCluster.java file to sort the list.

A while loop and switch method are implemented to create a function that parses Server commands such as JOBN and JOBP commands. Functions that retrieve DATA streams of servers that are capable of fulfilling the jobs and schedule the jobs to the largest servers are done using ‘if’ statements and messages sent to the severs through the Connection.java file.

## Reference

https://github.com/JoelDuff999/COMP3100