# Distributed Systems Job Scheduler Optimising Cost

Jarrod Adair - #45432260

## Introduction

This distributed systems project, focusing on stage 2, is the continuation of our work on the client by building a new scheduling algorithm, made in the Java language that is compatible with the ds-sim server simulator provided. This required us to build a job scheduler client that could schedule jobs via discrete sequences of commands sent over a communication layer, with the goal to dispatch every job in a cost-effective manner to try and beat any of the three base algorithms.

## Problem Definition

the description of scheduling problem and the definition of your objective function including the justification of your choice.

The scheduling problem that was outlined to us in stage 2 is that when optimising performance based around three objects:

- Minimisation of average turnaround time

- Maximisation of average resource utilization

- Minimisation of total server rental cost

There will be sacrifices in other metrics and as such part of the stage 2 problem is to make a decision on what metric to improve.

In this stage I will be focusing on **minimizing the total server rental cost metric** through a means of a customized scheduling algorithm to work in conjunction with the vanilla client my team and I built in stage 1. I made the decision to optimise for this specific metric because I felt like this one would be the easiest to reduce, having all server costs provided during runtime. I also hypothesized that reducing server cost would also cross over into the two other metrics as reducing the time servers were running would reduce costs as well as higher resource utilization which would allow for less servers to be required.

As showed in the figure above the distributed system has six components.

## Algorithm Description

You need to provide a simple example scheduling scenario including a sample configuration, the schedule, the description and discussion; this is to visualise how your scheduling algorithm works.

## Implementation

My algorithm was implemented in Java version 15 which is the latest version of the java development kid and the same version used in stage 1 of the project. Specifically, it was developed using the Adopt JDK, which is an open-source implementation of the Java development kit. I used Microsoft’s Visual Studio Code to develop the entirety of the algorithm and further client. The client was tested on Ubuntu 20.04 LTS in a VirtualBox virtual machine.

I specifically chose Java 15 not just because it was good practice as its becoming the new standard, overtaking java 8, but because of several language features that are simply not available on previous versions of Java. A notable example is the enhanced switch statement used heavily throughout the project, being available since Java 14. This enhanced switch statement was only one of the many new Java standard library improvements with other components such as TCP socket connections, XML parsing and I/O handling all included in the standard library; allowing for no third-party libraries to be included in the project.

Continuing with the implementation, I will be discussing files that received significant changes from stage 1 of the client as well as the new class files that were created to fulfill the assignment specifications of stage 2.

#### Client Class and Main Method

## Evaluation

s

## References

GitHub Repo: https://github.com/jarrod10/DS-Sim-Client.git

[1] Y. Lee, Y. Kim, and J. King, ‘ds-sim: A Distributed Systems Simulator User Guide’. [Online]. Available: https://github.com/distsys-MQ/ds-sim.

[2]