COMP3100 Distributed System: Stage 2

Introduction

The implementation of Stage 2 is to design and implement a new scheduling algorithm that completes the tasks given on this assignment. These are the objectives that the algorithm must achieve based on the following:

- Minimisation of average turnaround time
- Maximisation of average resource utilisation
- Minimisation of total server rental cost

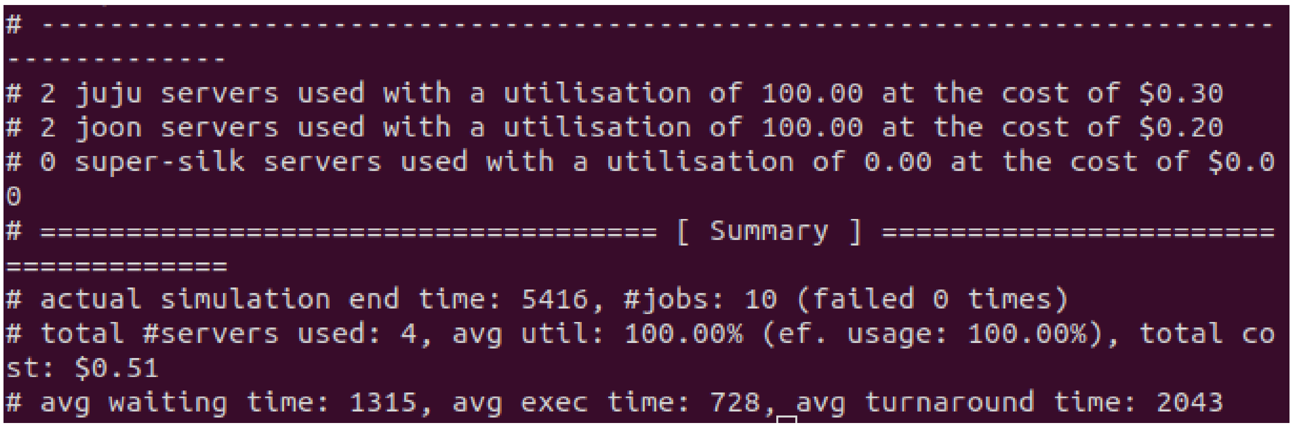
On the other hand, it is impossible to fulfil all the objectives when implementing an algorithm as if you prioritise to reduce the average turnaround time, the other two objectives will do the opposite. So I decided to implement an algorithm that reduces the average of total server rental cost by using ArrayList function.

Problem Statement

Understanding three algorithms like First Fit, Worst Fit and Best Fit gives a fundamental part to implement a custom algorithm. According to the Assignment 2, I have been assigned to implement the current algorithm from Stage 1 or create a new algorithm. As a result, I decided to create a new algorithm that can achieve one of the three objectives stated above. I created ‘lowCost’ algorithm which minimises the total rental cost.

Algorithm Description

The lowCost algorithm will create an object to store the details, and adding the new servers to the arraylist. It will choose and assign the most capable server in which it will compare the difference between the Server’s core and Job’s core that are required for the specific job. It will also make other comparison in terms of memory, disk, startTime and runTime. Once the comparison is done, it will schedule a new job in an arraylist.



```
# -------------------------------------------------------------------------
-------------
# 2 juju servers used with a utilisation of 100.00 at the cost of $0.30
# 2 joon servers used with a utilisation of 100.00 at the cost of $0.20
# 0 super-silk servers used with a utilisation of 0.00 at the cost of $0.0
0
# ==================================== [ Summary ] ======================
=============
# actual simulation end time: 5416, #jobs: 10 (failed 0 times)
# total #servers used: 4, avg util: 100.00% (ef. usage: 100.00%), total co
st: $0.51
# avg waiting time: 1315, avg exec time: 728, avg turnaround time: 2043
```

I have tested ds-sim by executing the ds-server ('./ds-server -c configs/sample-configs/ds-sample-config01.xml -v all -n') using the lowCost algorithm shown above. The total cost is $0.51 which it's the same, but it is more efficient when comes to the second test in Stage 2.

After running a few tests and analysing the results based on the terminal, we can see that the average waiting and average turnaround time are lower when implementing Stage 2 compared to Stage 1 algorithm. It is very beneficial that it improves in reducing the server rental cost overall and surprisingly, it reduces the average waiting time and average turnaround time as well. Therefore, the new algorithm provides better and affordable rental cost with available resources which utilises very well when assigning and scheduling jobs.

Implementation

Implementing new algorithm requires to configure Client.java, Server.java and Job.java in terms of classes. While implementing the new algorithm, I established a public connection between the libraries and the data streams by assigning the public static hostName and serverPort as "localhost" and "50000" respectively to ensure no human error, and it is able to send and receive messages between the client and the server. It is a two-way communication between the server and the client in which a socket will be created before connecting to the port.

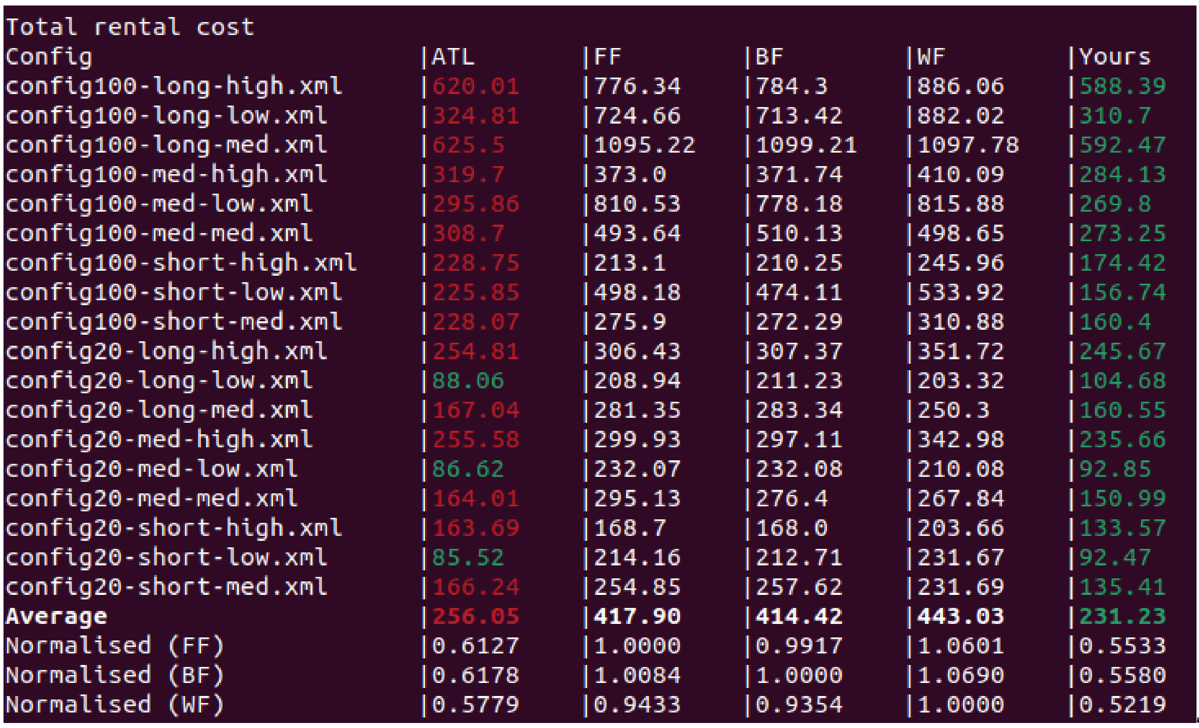
How it works is the server will listen and respond to the client whoever has the specific port number which is 50000, including the localhost as well. Once the client receives the respond from the server, it will also create the socket to match with the server’s socket, including the localhost in order to connect. Therefore, the connection is established.

Based on the lowCost algorithm, it will be setting up localhost and port number. Then, the algorithm will start with the HELO if the server is listening or not. When the server responds, it will exchange data based on read and send functions, including the lowCost algorithm. Once the server and client are acknowledged, it will run the test results given by the Assignment 2 binary test file, and will run the lowCost algorithm whether it meets the criteria or not. If the criteria is met, it will schedule the given job on the client side to the particular server. Otherwise, it will terminate an the scheduling if the criteria is not met. As a result, the loop will start again for another job until there is no jobs left. When the job is complete, it will print out, “JOBN” which means that it contains job details before sending it to the specific server.

The client will request the list of servers with “GETS Capable” to obtain and store data in an arraylist servers. This shows that the algorithm will compare the difference between server’s core and client’s core, it will assign the suitable server to fit which reduces the average rental cost.

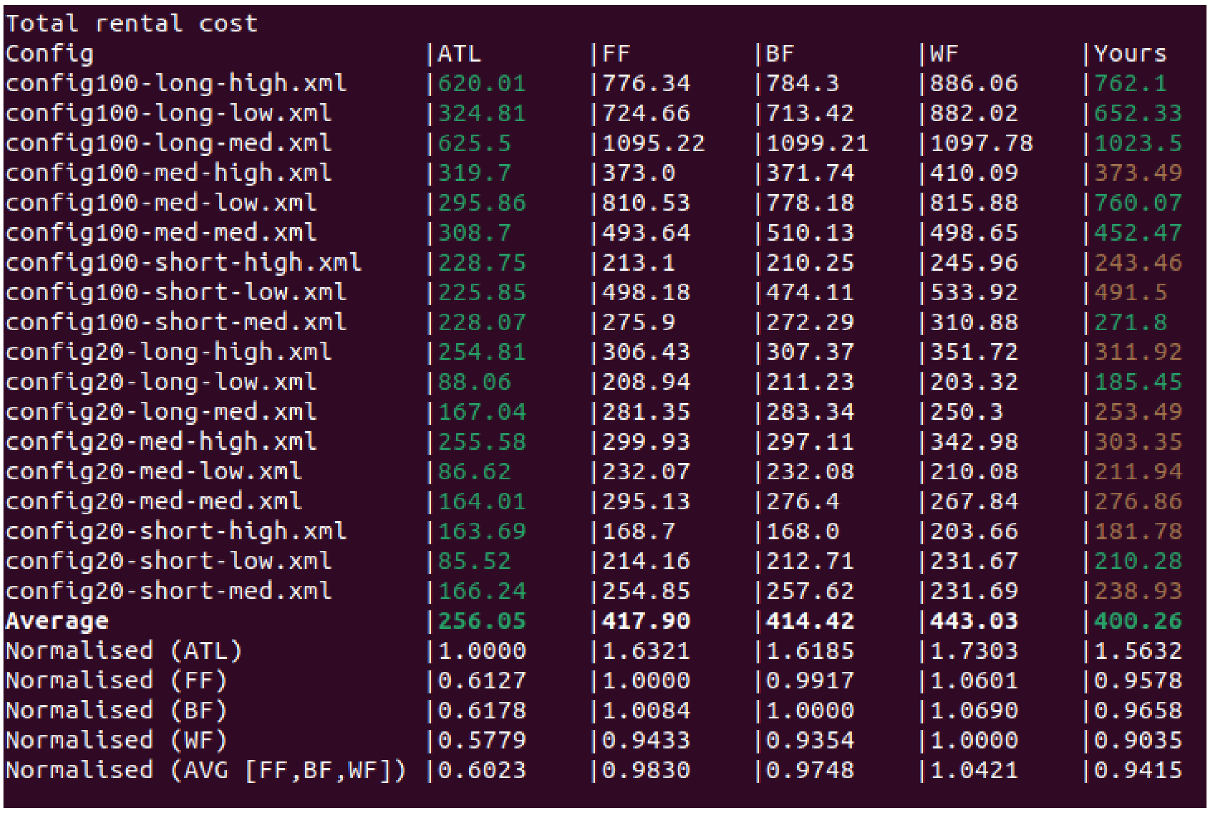
Evaluation

When I executed the algorithm with ds-server and did some analysing when running the configuration files (/configs/sample-configs). According to the tables shown below:



```
Total rental cost
Config                      |ATL       |FF        |BF        |WF        |Yours
config100-long-high.xml     |620.01    |776.34    |784.3     |886.06    |588.39
config100-long-low.xml      |324.81    |724.66    |713.42    |882.02    |310.7
config100-long-med.xml      |625.5     |1095.22   |1099.21   |1097.78   |592.47
config100-med-high.xml      |319.7     |373.0     |371.74    |410.09    |284.13
config100-med-low.xml       |295.86    |810.53    |778.18    |815.88    |269.8
config100-med-med.xml       |308.7     |493.64    |510.13    |498.65    |273.25
config100-short-high.xml    |228.75    |213.1     |210.25    |245.96    |174.42
config100-short-low.xml     |225.85    |498.18    |474.11    |533.92    |156.74
config100-short-med.xml     |228.07    |275.9     |272.29    |310.88    |160.4
config20-long-high.xml      |254.81    |306.43    |307.37    |351.72    |245.67
config20-long-low.xml       |88.06     |208.94    |211.23    |203.32    |104.68
config20-long-med.xml       |167.04    |281.35    |283.34    |250.3     |160.55
config20-med-high.xml       |255.58    |299.93    |297.11    |342.98    |235.66
config20-med-low.xml        |86.62     |232.07    |232.08    |210.08    |92.85
config20-med-med.xml        |164.01    |295.13    |276.4     |267.84    |150.99
config20-short-high.xml     |163.69    |168.7     |168.0     |203.66    |133.57
config20-short-low.xml      |85.52     |214.16    |212.71    |231.67    |92.47
config20-short-med.xml      |166.24    |254.85    |257.62    |231.69    |135.41
Average                     |256.05    |417.90    |414.42    |443.03    |231.23
Normalised (FF)             |0.6127    |1.0000    |0.9917    |1.0601    |0.5533
Normalised (BF)             |0.6178    |1.0084    |1.0000    |1.0690    |0.5580
Normalised (WF)             |0.5779    |0.9433    |0.9354    |1.0000    |0.5219
```

Stage 1: Total Rental Cost



```
Total rental cost
Config                      |ATL       |FF        |BF        |WF        |Yours
config100-long-high.xml     |620.01    |776.34    |784.3     |886.06    |762.1
config100-long-low.xml      |324.81    |724.66    |713.42    |882.02    |652.33
config100-long-med.xml      |625.5     |1095.22   |1099.21   |1097.78   |1023.5
config100-med-high.xml      |319.7     |373.0     |371.74    |410.09    |373.49
config100-med-low.xml       |295.86    |810.53    |778.18    |815.88    |760.07
config100-med-med.xml       |308.7     |493.64    |510.13    |498.65    |452.47
config100-short-high.xml    |228.75    |213.1     |210.25    |245.96    |243.46
config100-short-low.xml     |225.85    |498.18    |474.11    |533.92    |491.5
config100-short-med.xml     |228.07    |275.9     |272.29    |310.88    |271.8
config20-long-high.xml      |254.81    |306.43    |307.37    |351.72    |311.92
config20-long-low.xml       |88.06     |208.94    |211.23    |203.32    |185.45
config20-long-med.xml       |167.04    |281.35    |283.34    |250.3     |253.49
config20-med-high.xml       |255.58    |299.93    |297.11    |342.98    |303.35
config20-med-low.xml        |86.62     |232.07    |232.08    |210.08    |211.94
config20-med-med.xml        |164.01    |295.13    |276.4     |267.84    |276.86
config20-short-high.xml     |163.69    |168.7     |168.0     |203.66    |181.78
config20-short-low.xml      |85.52     |214.16    |212.71    |231.67    |210.28
config20-short-med.xml      |166.24    |254.85    |257.62    |231.69    |238.93
Average                     |256.05    |417.90    |414.42    |443.03    |400.26
Normalised (ATL)            |1.0000    |1.6321    |1.6185    |1.7303    |1.5632
Normalised (FF)             |0.6127    |1.0000    |0.9917    |1.0601    |0.9578
Normalised (BF)             |0.6178    |1.0084    |1.0000    |1.0690    |0.9658
Normalised (WF)             |0.5779    |0.9433    |0.9354    |1.0000    |0.9035
Normalised (AVG [FF,BF,WF]) |0.6023    |0.9830    |0.9748    |1.0421    |0.9415
```

Stage 2: Total Rental Cost Time

According to the tables in comparison, it shows a huge significant improvement for Stage 2 than Stage 1. I ran the test_results file to make comparison.

In Stage 1, the average rental from Yours, 231.23 failed to go above the requirement from ATL which is 256.05, whereas for Stage 2, the average rental from Yours, 400.26 which is above the requirement from ATL.

When comes to First Fit, Best Fit, and Worst Fit, both Stage 1 and Stage 2 didn't exceed the requirement. However, Stage 2 showed some results that indicated yellow in colour, but it showed better results when running the test.

Conclusion

In conclusion, the lowCost algorithm shows and performs better task in terms of reducing total rental cost and average rental cost. It even reduces the average waiting time and turnaround time when comparing Stage 1 and Stage 2. I can foresee that the algorithm can be improved to increase its efficiency, however, there is limited time to fulfil other two objectives.

Reference

https://github.com/IsaacZJMQ/COMP3100Stage2