Performance-efficient job scheduling algorithm for distributed systems

Randle Pete Valerio (45952507)

Introduction

The increasing use of data centres in today's society has introduced problems regarding the efficiency and effectiveness of the data centres such as poor speed and performance, poor resource utilisation and high rental costs. The efficiency and effectiveness of data centres can be seen in three objectives.

- One of the objectives is maximising the performance of the servers in the data centres.
- The second objective is maximising the utilisation of the resources of the servers in the data centres.
- The last objective is the minimisation of the rental cost of the servers that is being used.

Although it is possible to achieve one or two out of the three objectives it is impossible to achieve all three objectives as some of the objectives are contradicting each other. For example, to maximise the performance of the data centres, it will need to use more servers but using more servers means that it will increase the rental costs of the data centres. The scheduling algorithm that is to be designed in this project will be designed to achieve the objective of maximising the performance of the data centres, also keeping in mind the utilisation of the resources and the rental costs of the servers usage.

Problem Definition

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

The scheduling problem is being able to get the job, find out the required resources such as the core count, memory and disk and to be able to schedule the job to the server that meets the required resources criteria and do so in a way that minimises the turnaround time which is the time the job is submitted and then the time it gets to complete the job. Another scheduling problem is the utilisation of the resources such as the servers, making sure that the resources are being used well at mostly 100% of their capacity. The last scheduling problem is the reduction of the rental cost of the resources which means the reduction of the resources being used as the rental costs are being charged for the number of servers and resources that are being used.

This project’s objective function is to be able to minimise the turnaround time of scheduling all the jobs to the servers that meets it’s resource requirements and by doing so will maximise the performance of the distributed systems, but the higher the performance means that more servers will be used which also increases the rental costs of the distributed systems. However, people in this day and age value higher performance over costs, people would pay more just to be able to get better performance and speed.

Algorithm Description

Sample Configuration

```
<config randomSeed="1024">
  <servers>
    <server type="juju" limit="2" bootupTime="60" hourlyRate="0.2"
    coreCount="2" memory="4000" disk="16000"/>
    <server type="joon" limit="2" bootupTime="60" hourlyRate="0.4"
    coreCount="4" memory="16000" disk="64000"/>
    <server type="super-silk" limit="1" bootupTime="80" hourlyRate="0.8"
    coreCount="16" memory="64000" disk="512000"/>
  </servers>
  <jobs>
    <job type="short" minRunTime="1" maxRunTime="300" populationRate="60"/>
    <job type="medium" minRunTime="3011" maxRunTime="1800"
    populationRate="30"/>
    <job type="long" minRunTime="1801" maxRunTime="10000"
    populationRate="10"/>
  </jobs>
  <workload type="moderate" minLoad="30" maxLoad="70"/>
  <termination>
    <condition type="endtime" value="604800"/>
    <condition type="jobcount" value="10"/>
  </termination>
</config>
```

Scheduler

Description:

It first schedules the job to the very first server that has a little bit more resources than the required resources.

- Input:
  - A job
  - A array of servers, in the ascending order of the core capacity
- Output:
  - The server to schedule the **j** on
- Process:
  - It searches the list of servers from beginning to end and finds the server that best fits the criteria of:
    1. Servers with the required resources that are immediately available.
    2. Servers that has no jobs waiting
    3. Server that is better than the first server*
  - If there are no servers found that are immediately available, it then searches for capable servers that might be waiting/running or booting/active and in that list of capable servers it chooses the first server that has available resources that meets the required resources for the job.
  - *Server that is better than the first server means that, usually the first servers that are immediately available has a core count that exactly matches the required core count of the server, but this scheduler chooses a server that when the server's core count - the job's required core count is equal to 1 or more.

For instance, a job requiring a core count of 2, then it searches for a list of immediately available resources and found 5 with the first server juju having a core count of 2, the second server joon having a core count of 4, it will then proceed to choose the second server joon because the server juju will equal a 0 if the available core is subtracted by the required core but the second server joon will equal with a number equal or more than 1 which is 2.

  - If there is only one in the list it chooses the first one or if all the immediately available servers have the same core count it will choose the first one.

For example, if it a job needs a server with a core count of 2 and the search resulted in 2 servers juju id=0 with a core count of 2 and juju id=1 with a core count of 2, the schedule will choose juju id=0 as the server to schedule the job on.

Implementation

Inside the while loop is the location where the job scheduling part of the scheduler is implemented. In the while loop, the Scheduler sends a “REDY” command to the server indicating that it is ready to receive a job to be scheduled. The server then sends a job and the Scheduler stores the job in a string array called “jobn” and the string is split by a space/” “ and each split is put in different indexes in the array that is storing information about the job to be scheduled. It uses the first index in the john array to see if there are jobs that needs to be scheduled if john[0].equals(“NONE”) then it will break the while loop and terminates the connection between the Scheduler and the servers but if john[0].equals(“JOBN”) then the scheduler will run to search for a job the server needs.

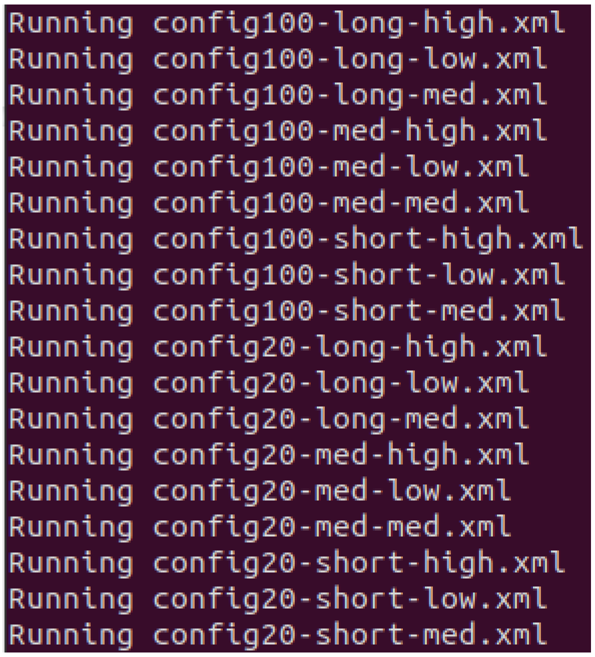
If job[0].equals(“JOBN”):
The Scheduler will send a GETS Avail command with the required resources that are stored in the job[4], job[5], and job[6] which are the core count, memory and disk. Server sends the DATA about the servers and the Scheduler stores it in an array and creates an integer variable to store the amount of servers that the DATA showed. The scheduler also created a variety of variables to store different server information and to create an indicator that when the right server is found the integer variable will tell the if statements that the rest won’t be needed.

If the DATA states that there are zero available servers then the scheduler will run a GETS Capable command with the required resources for the job. Then the scheduler will read and store the DATA the server sent in an array and the scheduler will send an OK message back to the server. After that, a for loop will be executed to search and choose the first capable server that also has available resources and schedules the job to the server that was chosen using the SCHD command.

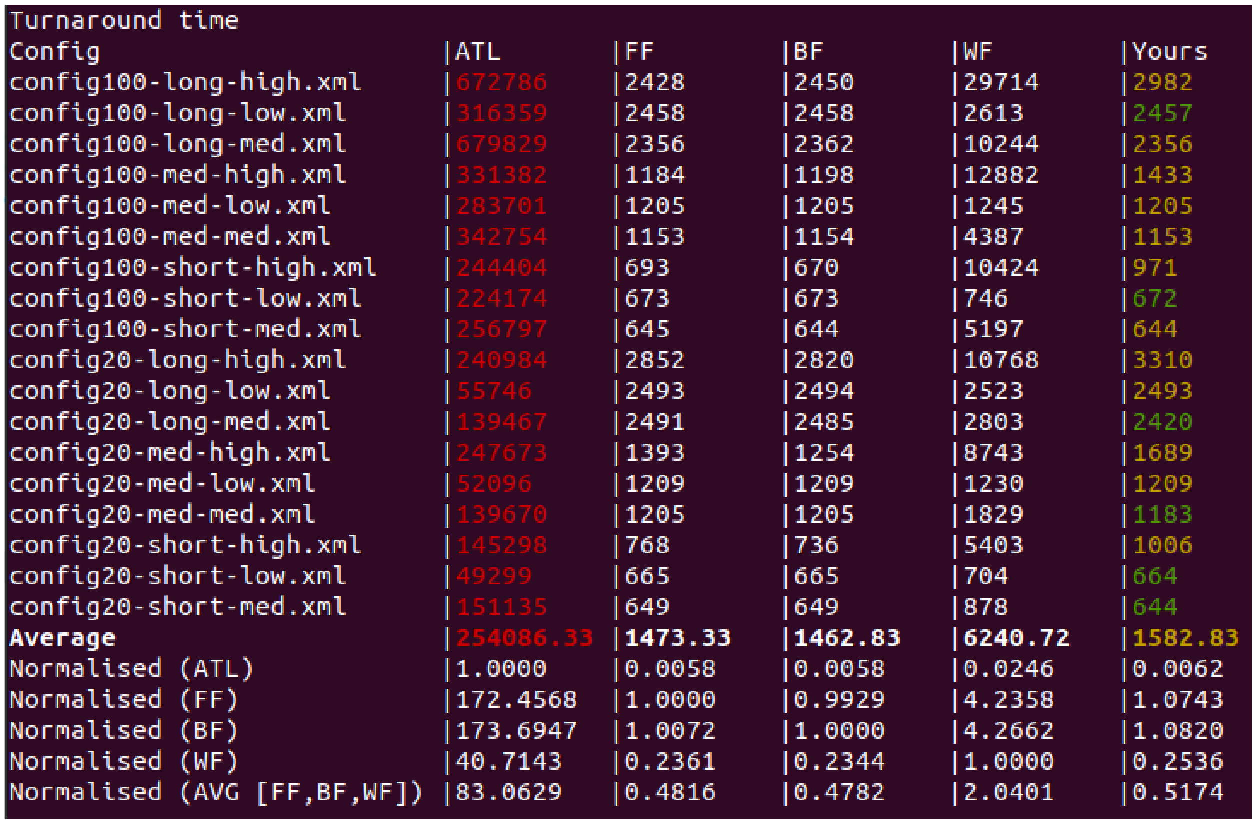
If the DATA is more than zero then it will continue using the information of the GETS Avail command. The scheduler uses a for loop to loop through all the immediately available servers that meet the required resources criteria. Using a variety of if statements to find the right server and storing the information needed such as the serverType and the serverID in the created string variables. The Scheduler then schedules the job using the SCHD command to the server that has the matching serverType and serverID with the ones stored in the schedulers variables.

Evaluation

Running the test_results file using the configurations in ds-sim-master/configs/other

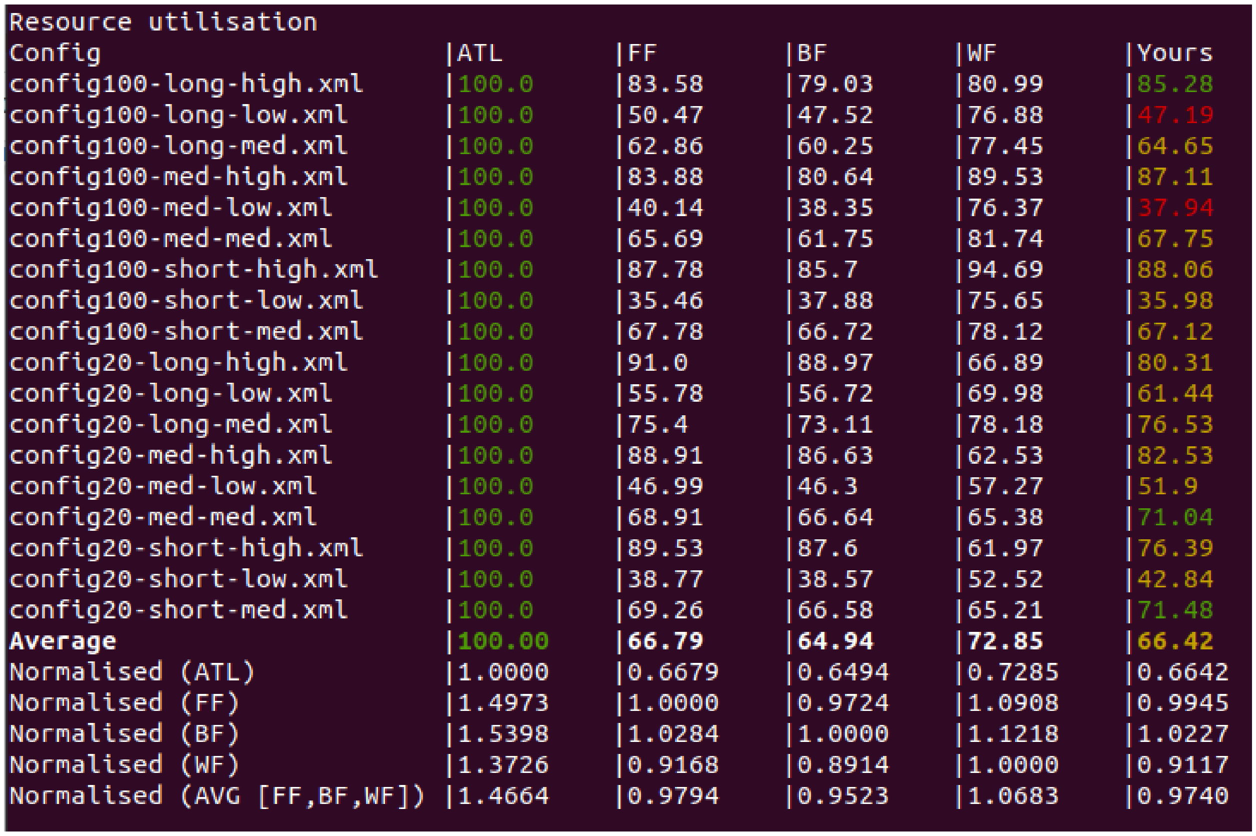


```
Running config100-long-high.xml
Running config100-long-low.xml
Running config100-long-med.xml
Running config100-med-high.xml
Running config100-med-low.xml
Running config100-med-med.xml
Running config100-short-high.xml
Running config100-short-low.xml
Running config100-short-med.xml
Running config20-long-high.xml
Running config20-long-low.xml
Running config20-long-med.xml
Running config20-med-high.xml
Running config20-med-low.xml
Running config20-med-med.xml
Running config20-short-high.xml
Running config20-short-low.xml
Running config20-short-med.xml
```



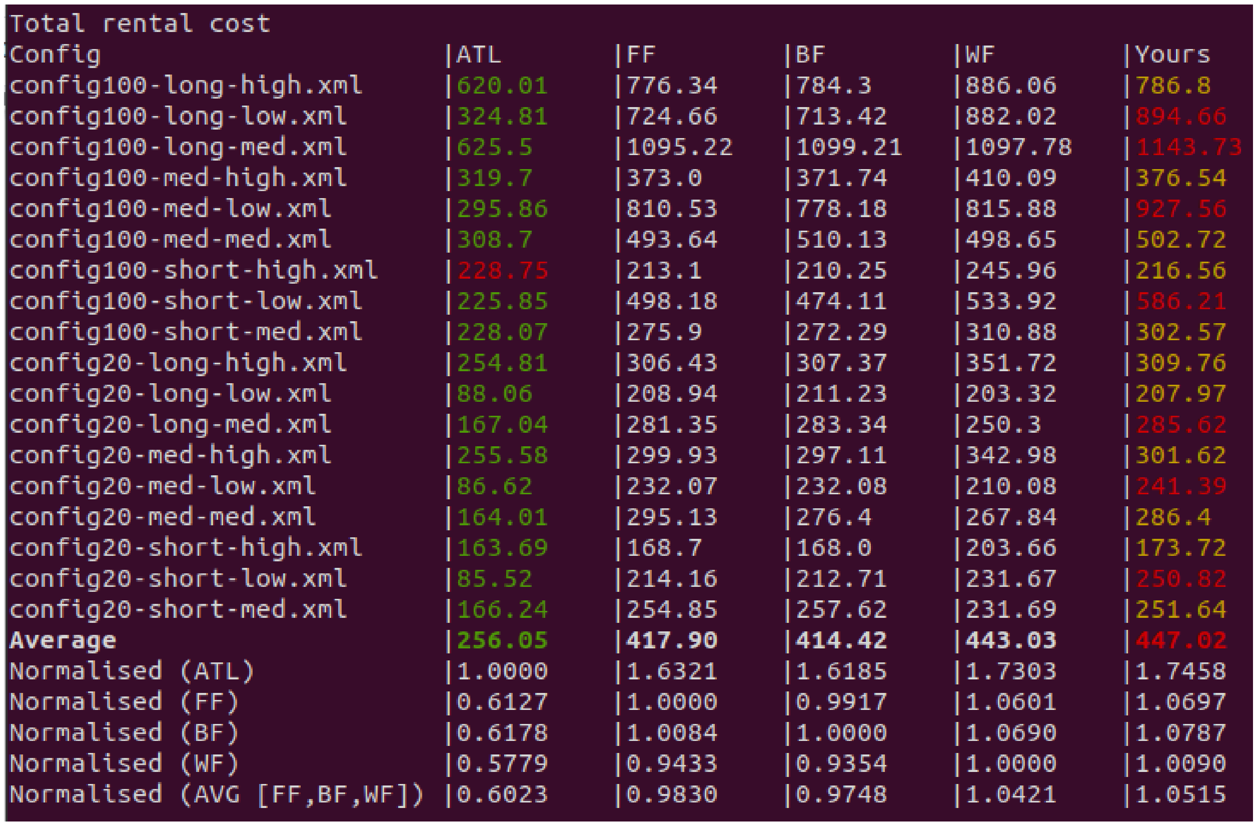
```
Turnaround time
Config                      |ATL       |FF        |BF        |WF        |Yours
config100-long-high.xml     |672786    |2428      |2450      |29714     |2982
config100-long-low.xml      |316359    |2458      |2458      |2613      |2457
config100-long-med.xml      |679829    |2356      |2362      |10244     |2356
config100-med-high.xml      |331382    |1184      |1198      |12882     |1433
config100-med-low.xml       |283701    |1205      |1205      |1245      |1205
config100-med-med.xml       |342754    |1153      |1154      |4387      |1153
config100-short-high.xml    |244404    |693       |670       |10424     |971
config100-short-low.xml     |224174    |673       |673       |746       |672
config100-short-med.xml     |256797    |645       |644       |5197      |644
config20-long-high.xml      |240984    |2852      |2820      |10768     |3310
config20-long-low.xml       |55746     |2493      |2494      |2523      |2493
config20-long-med.xml       |139467    |2491      |2485      |2803      |2420
config20-med-high.xml       |247673    |1393      |1254      |8743      |1689
config20-med-low.xml        |52096     |1209      |1209      |1230      |1209
config20-med-med.xml        |139670    |1205      |1205      |1829      |1183
config20-short-high.xml     |145298    |768       |736       |5403      |1006
config20-short-low.xml      |49299     |665       |665       |704       |664
config20-short-med.xml      |151135    |649       |649       |878       |644
Average                     |254086.33 |1473.33   |1462.83   |6240.72   |1582.83
Normalised (ATL)            |1.0000    |0.0058    |0.0058    |0.0246    |0.0062
Normalised (FF)             |172.4568  |1.0000    |0.9929    |4.2358    |1.0743
Normalised (BF)             |173.6947  |1.0072    |1.0000    |4.2662    |1.0820
Normalised (WF)             |40.7143   |0.2361    |0.2344    |1.0000    |0.2536
Normalised (AVG [FF,BF,WF]) |83.0629   |0.4816    |0.4782    |2.0401    |0.5174
```

The Scheduling Algorithm’s turnaround time results are significantly better in comparison with the WF/Worst Fit Algorithm. For the BF/Best Fit algorithm comparison with the Scheduling algorithm the results are a spreaded in a way that some of them are better, some of them have the same results and a few of them have worse results. The Scheduling algorithm’s comparison with the FF/First Fit algorithm are somewhat the same as the BF/Best Fit in the aspect of the results having varying comparisons such as some of them are better, the same, and a few worse cases.



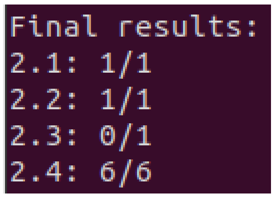
```
Resource utilisation
Config                      |ATL        |FF         |BF         |WF         |Yours
config100-long-high.xml     |100.0      |83.58      |79.03      |80.99      |85.28
config100-long-low.xml      |100.0      |50.47      |47.52      |76.88      |47.19
config100-long-med.xml      |100.0      |62.86      |60.25      |77.45      |64.65
config100-med-high.xml      |100.0      |83.88      |80.64      |89.53      |87.11
config100-med-low.xml       |100.0      |40.14      |38.35      |76.37      |37.94
config100-med-med.xml       |100.0      |65.69      |61.75      |81.74      |67.75
config100-short-high.xml    |100.0      |87.78      |85.7       |94.69      |88.06
config100-short-low.xml     |100.0      |35.46      |37.88      |75.65      |35.98
config100-short-med.xml     |100.0      |67.78      |66.72      |78.12      |67.12
config20-long-high.xml      |100.0      |91.0       |88.97      |66.89      |80.31
config20-long-low.xml       |100.0      |55.78      |56.72      |69.98      |61.44
config20-long-med.xml       |100.0      |75.4       |73.11      |78.18      |76.53
config20-med-high.xml       |100.0      |88.91      |86.63      |62.53      |82.53
config20-med-low.xml        |100.0      |46.99      |46.3       |57.27      |51.9
config20-med-med.xml        |100.0      |68.91      |66.64      |65.38      |71.04
config20-short-high.xml     |100.0      |89.53      |87.6       |61.97      |76.39
config20-short-low.xml      |100.0      |38.77      |38.57      |52.52      |42.84
config20-short-med.xml      |100.0      |69.26      |66.58      |65.21      |71.48
Average                     |100.00     |66.79      |64.94      |72.85      |66.42
Normalised (ATL)            |1.0000     |0.6679     |0.6494     |0.7285     |0.6642
Normalised (FF)             |1.4973     |1.0000     |0.9724     |1.0908     |0.9945
Normalised (BF)             |1.5398     |1.0284     |1.0000     |1.1218     |1.0227
Normalised (WF)             |1.3726     |0.9168     |0.8914     |1.0000     |0.9117
Normalised (AVG [FF,BF,WF]) |1.4664     |0.9794     |0.9523     |1.0683     |0.9740
```

The Scheduling Algorithm's resource utilisation results have some that are marginally better compared to the three scheduling algorithms ff, bf, and wf and there's two that are doing worse than the three scheduling algorithms and most of them are in the average zone of the three scheduling algorithms.



```
Total rental cost
Config                         |ATL        |FF         |BF         |WF         |Yours
config100-long-high.xml        |620.01     |776.34     |784.3      |886.06     |786.8
config100-long-low.xml         |324.81     |724.66     |713.42     |882.02     |894.66
config100-long-med.xml         |625.5      |1095.22    |1099.21    |1097.78    |1143.73
config100-med-high.xml         |319.7      |373.0      |371.74     |410.09     |376.54
config100-med-low.xml          |295.86     |810.53     |778.18     |815.88     |927.56
config100-med-med.xml          |308.7      |493.64     |510.13     |498.65     |502.72
config100-short-high.xml       |228.75     |213.1      |210.25     |245.96     |216.56
config100-short-low.xml        |225.85     |498.18     |474.11     |533.92     |586.21
config100-short-med.xml        |228.07     |275.9      |272.29     |310.88     |302.57
config20-long-high.xml         |254.81     |306.43     |307.37     |351.72     |309.76
config20-long-low.xml          |88.06      |208.94     |211.23     |203.32     |207.97
config20-long-med.xml          |167.04     |281.35     |283.34     |250.3      |285.62
config20-med-high.xml          |255.58     |299.93     |297.11     |342.98     |301.62
config20-med-low.xml           |86.62      |232.07     |232.08     |210.08     |241.39
config20-med-med.xml           |164.01     |295.13     |276.4      |267.84     |286.4
config20-short-high.xml        |163.69     |168.7      |168.0      |203.66     |173.72
config20-short-low.xml         |85.52      |214.16     |212.71     |231.67     |250.82
config20-short-med.xml         |166.24     |254.85     |257.62     |231.69     |251.64
Average                        |256.05     |417.90     |414.42     |443.03     |447.02
Normalised (ATL)               |1.0000     |1.6321     |1.6185     |1.7303     |1.7458
Normalised (FF)                |0.6127     |1.0000     |0.9917     |1.0601     |1.0697
Normalised (BF)                |0.6178     |1.0084     |1.0000     |1.0690     |1.0787
Normalised (WF)                |0.5779     |0.9433     |0.9354     |1.0000     |1.0090
Normalised (AVG [FF,BF,WF])    |0.6023     |0.9830     |0.9748     |1.0421     |1.0515
```

Due to the increase in servers being used in the Scheduling algorithm, it can be seen that there is also an increase in the rental costs for the results of the scheduling algorithm making almost half of the results worse or in the red compared to the three scheduling algorithms ff, bf, and wf.



```
Final results:
2.1: 1/1
2.2: 1/1
2.3: 0/1
2.4: 6/6
```

The Scheduling algorithm although slightly increased the performance of the turnaround time by using more servers which lead to a higher rental costs of the servers being used. Another factor to consider is the resource utilisation, because the implementation of the scheduling algorithm chooses the second level server in the list of the available servers some of the servers with large amounts of resources such as a core count of 8 are being scheduled with jobs that has a core count of 2 which results in a poor utilisation of resources as the servers aren't being used for their maximum capacity.

Pros:

- Have a slightly faster performance when compared to the turnaround time of the scheduling algorithm

Cons:

- The used of more servers garnered more rental costs
- Not being able to properly utilise the resources’ 100% capacity

Conclusion

It is impossible to be able to achieve all the three objectives in creating a scheduling algorithm because some of the objectives need to sacrifice another objective in order for it to progress such as the maximisation of the performance will mean a rise in the rental costs for the distributed systems. My suggestion is to prioritise the objective that you want to achieve while not degrading the other objectives in a huge way, try and to keep it a little more balanced.

References:

- Valerio, R., (2021) COMP3100-A2 [Source Code]. <https://github.com/RandleValerio/COMP3100-A2>
- Lee, Y.C., (2021) DS-Sim [Source Code]. <https://github.com/distsys-MQ/ds-sim>
- Lee, Y.C., Kim, Y.K., King, J., (2021) ds-sim: A Distributed Systems simulator User Guide