Project report: Stage 2
COMP3100 Distributed Systems, S2, 2022
SID: 45456070, Name: Beau Williams

1. Introduction

For this COMP3100 assignment we will discuss the process of designing and implementing a new job scheduling algorithm for the Distributed Systems Server Simulator (DSSIM)[1]. This report shall cover stage 2 of this assignment, which involves the implementation of the a new Scheduler Algorithm, which will be called First Available Capable Scheduler (FAC), as well as interfacing with the DSSIM to perform simulations with provided test criteria to determine the performance of this new algorithm.

The purpose of the report coupled with the design and implementation of the FAC Scheduler is intended to test our understanding of Distributed System in relation to the performance tradeoffs of verious scheduling algorithms. It will be made clear that there is no silver bullet algorithm that will perform more efficiently in every aspect (cost, turnaround time, utilisation). As a result, in order to design an effective algorithm one must make a choice as to which aspect of performance they wish to optimise for. Design complexity is not inherently a solution to this optimisation dilemma either. By simply adding more features to an algorithm not only does its characteristics change, but also it often becomes harder to maintain. While there might be some optimisations that inch out slightly better algorithm performance in all aspects, it's still not fundamentally a better design, given that it is still the basic implementation but with some added optimisations. As the old saying goes, less is often more. The design of the FAC scheduler is purposefully simple, albeit at the cost of a slightly higher operating cost (and design complexity), however it does improve the turnaround time which is the very goal of this implementation.

2. FAC Scheduler Overview

At a high level, the job scheduling system that we are creating adopts a similar approach to the Largest Round Robin (LRR) and First Fit (FF) algorithms. It has been designed to blend the turnaround time efficiency of FF algorithms and the utilisation efficiency of the LRR algorithm to create an algorithm the exhibits greater turnaround time efficiency over the baseline algorithms First Fit, Best Fit, Worst Fit and First Capable.

3. Problem Definition

Here we discuss the problem that the First Available Capable algorithm seeks to solve, namely we wish to optimise for turnaround time, while not significantly increasing rental cost and decreasing resource utilisation efficiency.

This problem can be described as the optimisation trilemma that is made up of a combination of the three hey performance characteristics of a scheduling algorithm. Namely, rental cost, resource utilisation and turnaround time.

At best a scheduling algorithm designer can pick two characteristics to optimise for, at the expense of the third not chosen characteristic. It is evident that there is a direct correlation to performance improvements to one characteristic, which shall negatively affect the performance of another characteristic.

4. Algorithm Description

First Avaailable Capable Scheduler is designed to improve upon the baseline algorithm First Fit by combining Largest Round Robin as a fall back scheduler when there are no available servers. The key difference between First Fit and First Available Capable is that FAC is designed to evenly distribute jobs to busy servers when there are non available, explicitly preferring larger server types.

In regards to the problem description, it has been chosen to improve the turnaround time of this algorithm, while not causing a significant detriment to the other two remaining characteristics, rental cost and resource utilisation.

4.1. Last Round Robin

Last Round Robin (LRR) is designed to dispatch jobs to the largest servers available. It determines the largest servers by their core count, and distributes jobs evenly to those servers in a round robin fashion. Initially it will begin with the first server located from the get servers result returned from the DSSIM. Once the first job has been dispatched it will then proceed to increment a counter, dispatching jobs to each server one by one until it reaches the last server. At this point it will start from the first server and repeat this cycle until there are no more remaining jobs to dispatch. This method exhibits moderately efficient resource utilisation and turnaround time are the expense of rental cost.

4.2. First Fit

First Fit (FF) is an algorithm that is designed to dispatch jobs to the first available server, that is also capable or running the job. It works of a simple assumption that if a job was dispatched to a server it will no longer be available (likely) to receive a job when dispatching the next job. This simple design means that first fit constantly targets the first server it finds that is available. Once available servers are exhausted, it will simply dispatch jobs to the first server it finds. FF is a simple implementation that has low code complexity. It has an optimised turnaround time but this comes at the detriment of rental cost and resource utilisation. It can be noted that first fit under performs in scenarios where there are no available servers are compared to LRR.

4.2. First Available Capable

First Available Capable (FAC) is designed to be an elegant algorithm that is a hybrid of LRR and FF. Given that the problem definition states that we wish to maximise for turnaround time efficiency, we choose to improve upon the FF algorithm which has the greatest turnaround time efficiency of the baseline algorithms (FF, WF, BF and FC). FAC works by storing a list of capable servers upon first run on the client side. This list is used as the fallback list where jobs are distributed to capable servers in a similar fashion to LRR. When servers are available to receive jobs FAC simply sends it to the first available server. When servers are not available, FAC is explicitly designed to evenly distribute jobs to the largest capable (busy) servers. This fallback technique ensures that for situations where there are a large amount of jobs and few servers, that the turnaround time performance is optimised. In scenarios where there are many servers available, FAC exhibits similar characteristics to FF in that it has an optimised turnaround time efficiency.

The FAC Scheduler Algorithm is shown at a high level by figure 1

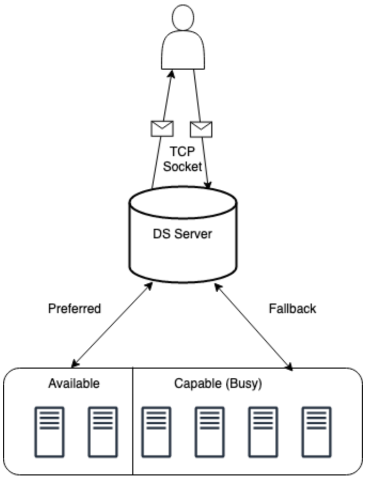


Figure 1: FAC Scheduler Architecture

5. Implementation

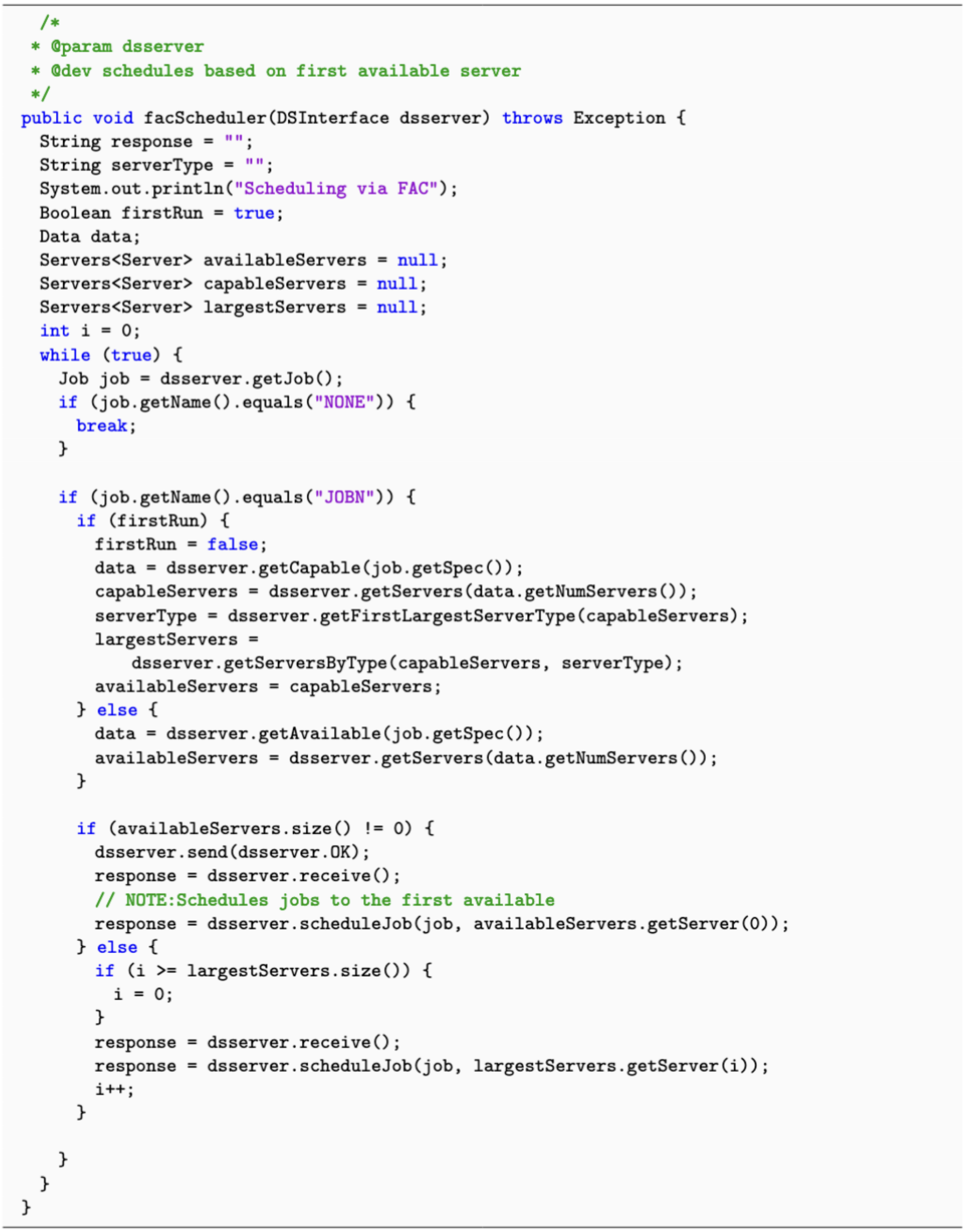
The implementation of the FAC Scheduling algorithm is designed to use minimal dependencies. Preferring to use well utilised data structures available in the standard Java library such as a Arraylists. A while loop is used for simplicity over a for loop to implement the core logic of the algorithm.

5.1. Tooling

This algorithm was programmed using a modal text editor called neovim[2], coupled with the the Ecplipse Language Server for Java (LSP)[3]

5.2. First Available Capable Algorithm (FAC)

The first available capable algorithm is designed to have a simple implementation while maximising for turnaround time efficiency.
The FAC Scheduler Algorithm is shown at a high level by figure 1



```java
  /*
 * @param dsserver
 * @dev schedules based on first available server
 */
public void facScheduler(DSInterface dsserver) throws Exception {
  String response = "";
  String serverType = "";
  System.out.println("Scheduling via FAC");
  Boolean firstRun = true;
  Data data;
  Servers<Server> availableServers = null;
  Servers<Server> capableServers = null;
  Servers<Server> largestServers = null;
  int i = 0;
  while (true) {
    Job job = dsserver.getJob();
    if (job.getName().equals("NONE")) {
      break;
    }

    if (job.getName().equals("JOBN")) {
      if (firstRun) {
        firstRun = false;
        data = dsserver.getCapable(job.getSpec());
        capableServers = dsserver.getServers(data.getNumServers());
        serverType = dsserver.getFirstLargestServerType(capableServers);
        largestServers =
            dsserver.getServersByType(capableServers, serverType);
        availableServers = capableServers;
      } else {
        data = dsserver.getAvailable(job.getSpec());
        availableServers = dsserver.getServers(data.getNumServers());
      }

      if (availableServers.size() != 0) {
        dsserver.send(dsserver.OK);
        response = dsserver.receive();
        // NOTE:Schedules jobs to the first available
        response = dsserver.scheduleJob(job, availableServers.getServer(0));
      } else {
        if (i >= largestServers.size()) {
          i = 0;
        }
        response = dsserver.receive();
        response = dsserver.scheduleJob(job, largestServers.getServer(i));
        i++;
      }

    }
  }
}
```

Figure 2: FAC Scheduler Implementation

6. Evaluation

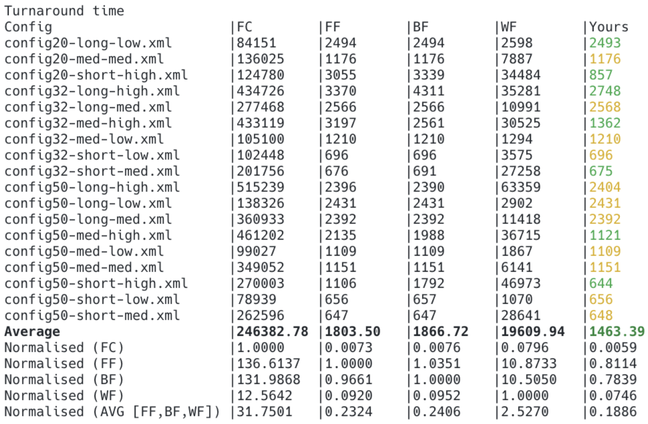
By analysing the optimisation trilemma faced by scheduling algorithms, turnaround time, rental cost and resource utilisation we start by analysing the cost of preferring one dimension over another. The FAC is designed to maximise on turnaround time efficiency, with good resource utilisation at the cost of rental cost.

6.1. Cost Benefits Analysis

Cost	Benefit
Increased Rental Cost	Turnaround Efficiency
Increased Rental Cost	Larger Jobs Processed Faster
Code Complexity	Optimised Algorithm
Slightly Reduced Utilisation	Turnaround Efficiency

6.2. Results

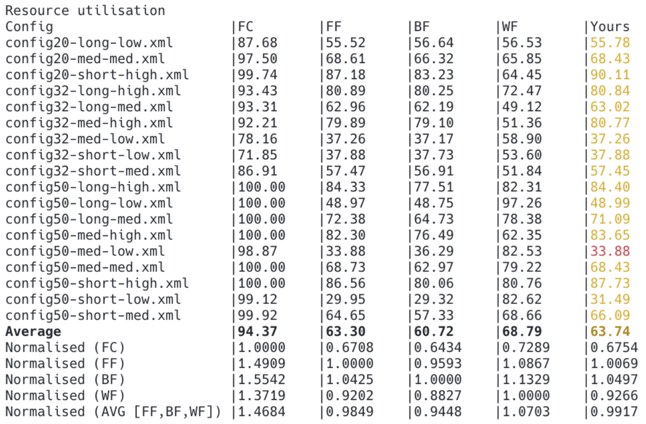
The FAC scheduler exhibits turnaround time efficiency 5



```
Turnaround time
Config                     |FC        |FF        |BF        |WF        |Yours
config20-long-low.xml      |84151     |2494      |2494      |2598      |2493
config20-med-med.xml       |136025    |1176      |1176      |7887      |1176
config20-short-high.xml    |124780    |3055      |3339      |34484     |857
config32-long-high.xml     |434726    |3370      |4311      |35281     |2748
config32-long-med.xml      |277468    |2566      |2566      |10991     |2568
config32-med-high.xml      |433119    |3197      |2561      |30525     |1362
config32-med-low.xml       |105100    |1210      |1210      |1294      |1210
config32-short-low.xml     |102448    |696       |696       |3575      |696
config32-short-med.xml     |201756    |676       |691       |27258     |675
config50-long-high.xml     |515239    |2396      |2390      |63359     |2404
config50-long-low.xml      |138326    |2431      |2431      |2902      |2431
config50-long-med.xml      |360933    |2392      |2392      |11418     |2392
config50-med-high.xml      |461202    |2135      |1988      |36715     |1121
config50-med-low.xml       |99027     |1109      |1109      |1867      |1109
config50-med-med.xml       |349052    |1151      |1151      |6141      |1151
config50-short-high.xml    |270003    |1106      |1792      |46973     |644
config50-short-low.xml     |78939     |656       |657       |1070      |656
config50-short-med.xml     |262596    |647       |647       |28641     |648
Average                    |246382.78 |1803.50   |1866.72   |19609.94  |1463.39
Normalised (FC)            |1.0000    |0.0073    |0.0076    |0.0796    |0.0059
Normalised (FF)            |136.6137  |1.0000    |1.0351    |10.8733   |0.8114
Normalised (BF)            |131.9868  |0.9661    |1.0000    |10.5050   |0.7839
Normalised (WF)            |12.5642   |0.0920    |0.0952    |1.0000    |0.0746
Normalised (AVG [FF,BF,WF])|31.7501   |0.2324    |0.2406    |2.5270    |0.1886
```

Figure 3: Turnaround Time Results

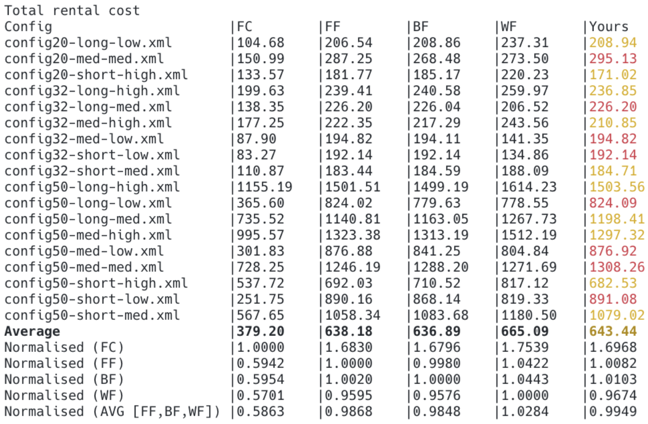
The FAC scheduler also exhibits moderate utilisation 5



```
Resource utilisation
Config                     |FC        |FF        |BF        |WF        |Yours
config20-long-low.xml      |87.68     |55.52     |56.64     |56.53     |55.78
config20-med-med.xml       |97.50     |68.61     |66.32     |65.85     |68.43
config20-short-high.xml    |99.74     |87.18     |83.23     |64.45     |90.11
config32-long-high.xml     |93.43     |80.89     |80.25     |72.47     |80.84
config32-long-med.xml      |93.31     |62.96     |62.19     |49.12     |63.02
config32-med-high.xml      |92.21     |79.89     |79.10     |51.36     |80.77
config32-med-low.xml       |78.16     |37.26     |37.17     |58.90     |37.26
config32-short-low.xml     |71.85     |37.88     |37.73     |53.60     |37.88
config32-short-med.xml     |86.91     |57.47     |56.91     |51.84     |57.45
config50-long-high.xml     |100.00    |84.33     |77.51     |82.31     |84.40
config50-long-low.xml      |100.00    |48.97     |48.75     |97.26     |48.99
config50-long-med.xml      |100.00    |72.38     |64.73     |78.38     |71.09
config50-med-high.xml      |100.00    |82.30     |76.49     |62.35     |83.65
config50-med-low.xml       |98.87     |33.88     |36.29     |82.53     |33.88
config50-med-med.xml       |100.00    |68.73     |62.97     |79.22     |68.43
config50-short-high.xml    |100.00    |86.56     |80.06     |80.76     |87.73
config50-short-low.xml     |99.12     |29.95     |29.32     |82.62     |31.49
config50-short-med.xml     |99.92     |64.65     |57.33     |68.66     |66.09
Average                    |94.37     |63.30     |60.72     |68.79     |63.74
Normalised (FC)            |1.0000    |0.6708    |0.6434    |0.7289    |0.6754
Normalised (FF)            |1.4909    |1.0000    |0.9593    |1.0867    |1.0069
Normalised (BF)            |1.5542    |1.0425    |1.0000    |1.1329    |1.0497
Normalised (WF)            |1.3719    |0.9202    |0.8827    |1.0000    |0.9266
Normalised (AVG [FF,BF,WF])|1.4684    |0.9849    |0.9448    |1.0703    |0.9917
```

Figure 4: Resource Utilisation Results

The FAC scheduler turnaround time efficiency comes at the detriment to cost 5



```
Total rental cost
Config                     |FC        |FF        |BF        |WF        |Yours
config20-long-low.xml      |104.68    |206.54    |208.86    |237.31    |208.94
config20-med-med.xml       |150.99    |287.25    |268.48    |273.50    |295.13
config20-short-high.xml    |133.57    |181.77    |185.17    |220.23    |171.02
config32-long-high.xml     |199.63    |239.41    |240.58    |259.97    |236.85
config32-long-med.xml      |138.35    |226.20    |226.04    |206.52    |226.20
config32-med-high.xml      |177.25    |222.35    |217.29    |243.56    |210.85
config32-med-low.xml       |87.90     |194.82    |194.11    |141.35    |194.82
config32-short-low.xml     |83.27     |192.14    |192.14    |134.86    |192.14
config32-short-med.xml     |110.87    |183.44    |184.59    |188.09    |184.71
config50-long-high.xml     |1155.19   |1501.51   |1499.19   |1614.23   |1503.56
config50-long-low.xml      |365.60    |824.02    |779.63    |778.55    |824.09
config50-long-med.xml      |735.52    |1140.81   |1163.05   |1267.73   |1198.41
config50-med-high.xml      |995.57    |1323.38   |1313.19   |1512.19   |1297.32
config50-med-low.xml       |301.83    |876.88    |841.25    |804.84    |876.92
config50-med-med.xml       |728.25    |1246.19   |1288.20   |1271.69   |1308.26
config50-short-high.xml    |537.72    |692.03    |710.52    |817.12    |682.53
config50-short-low.xml     |251.75    |890.16    |868.14    |819.33    |891.08
config50-short-med.xml     |567.65    |1058.34   |1083.68   |1180.50   |1079.02
Average                    |379.20    |638.18    |636.89    |665.09    |643.44
Normalised (FC)            |1.0000    |1.6830    |1.6796    |1.7539    |1.6968
Normalised (FF)            |0.5942    |1.0000    |0.9980    |1.0422    |1.0082
Normalised (BF)            |0.5954    |1.0020    |1.0000    |1.0443    |1.0103
Normalised (WF)            |0.5701    |0.9595    |0.9576    |1.0000    |0.9674
Normalised (AVG [FF,BF,WF])|0.5863    |0.9868    |0.9848    |1.0284    |0.9949
```

Figure 5: Rental Cost Results

Conclusion

This report has highlighted the inherent tradeoffs involved with designing optimised and efficient algorithms to schedule jobs to servers within a distributed systems environment. We have discussed the trilemma of 1. Turnaround Time 2. Resource Utilisation and 3. Rental Cost. It is clear that there is no one-size-fits-all approach to designing an algorithm. Where potentially there might be optimisations to further improve an algorithm, which may even improve the performance of it on all three basis points, it will still exhibit some performance trade-offs in some basis points as compared to other algorithms. To design an effective algorithm to schedule jobs to the DSSIM server, one must first decide which characteristics they desire from the algorithm. Whether it be effective utilisation of resources, low rental cost or fast turnaround time.

Source Code Repository

- https://github.com/beauwilliams/Comp3100

References

[1] Y. C. Lee, "Ds-sim." `https://github.com/distsys-MQ/ds-sim`, 2022.

[2] B. Williams, "Neovim," 2022.

[3] "Language server protocol."