# Job Scheduler

For Distributed Systems

Cong Trinh
SID: 44580460

# Table of Contents



# Introduction

The ds-sim is implemented based on discrete-event simulation, which simulates a system whereby each event happens instantaneously, and system states do not change between events. There are two components in the model: ds-client and ds-server. The ds-server simulates job submissions and job execution while the ds-client's functionality is a job scheduler.

The stage 1's implementation of the scheduler was based on the All-To-Largest (ALT) algorithm, which performs efficiently in maximising the utility's resources and minimising the cost. There are also three basic algorithms: best-fit (BF), first-fit (FF) and worst-fit (WF). The goal of stage 2 is to design an improved algorithm that will outperform the existing algorithms based on the following metrics:

- Minimasation of average turnaround time
- Maximasation of  average resource utilisation
- Minimastation of total server rental cost

# Problems

A problem for the job scheduler is that there is no optimal algorithm that can satisfy all the above metrics. In case that the scheduling algorithm is designed to minimise the rental cost, the turnaround time is always significantly high. Vice versa, if the algorithm minimises the turnaround time, the rental cost is almost always high compared to other algorithms.

The objective of the new algorithm is to improve each metric compared to other algorithms rather than optimising the schedule for one metric only. The performance that is expected from this algorithm is the followings:

- Average turnaround time:
    - Significantly better than ALT algorithm
    - Slightly worse or equal to FF, BF and WF
- Average resource utilisation:
    - Significantly better than FF, BF
    - Slightly better than WF
    - Cannot compete with ALT as it is always 100%
- Average rental cost:
    - Worse than ALT
    - Slightly better than FF, BF, WF

# Algorithms

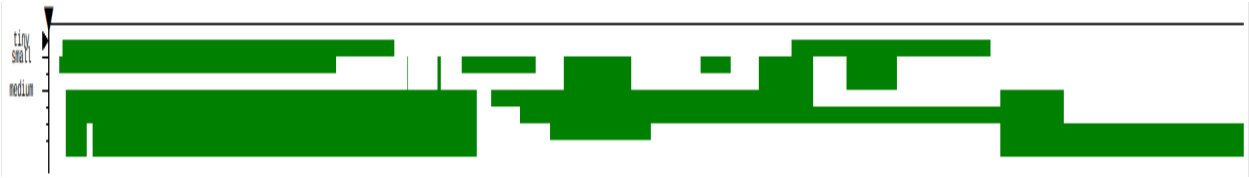
The algorithm is based on the first-fit approach. The scheduler always tries to assign a job to the first sufficient resource readily available server. If there is no available server, the algorithm starts to look for servers that can still have resources (i.e number of cores left from the scheduled jobs or running jobs) so that the job can be run parallel to the previous jobs. A details of the algorithm can be found in the followings:

1. Look for the first readily available server. Assign a job if a server is found, if not go to 2.
2. Look for the first server that satisfies below metrics. Return assigns the job immediately. Otherwise go to 3.
    a. There are waiting jobs/running jobs (i.e busy servers)
    b. Estimated finish time of assigned jobs is bigger than the submission time of the job
    c. The remaining resources (i.e. remaining cores) is sufficient to perform the job
3. Look for the server that has the earliest finish time or the server is cheaper to perform the job and assign the job to it.

Configs file is modified version of week 9 config file.

```
<config randomSeed="65535">
  <servers>
    <server type="tiny" limit="1" bootupTime="40" hourlyRate="0.4" coreCount="1"
    memory="4000" disk="32000"/>
    <server type="small" limit="1" bootupTime="40" hourlyRate="0.4"
    coreCount="2" memory="8000" disk="64000"/>
```

```
    <server type="medium" limit="1" bootupTime="60" hourlyRate="0.8"
    coreCount="4" memory="16000" disk="128000"/>
  </servers>
  <jobs>
    <job type="short" minRunTime="1" maxRunTime="60" populationRate="30"/>
    <job type="medium" minRunTime="61" maxRunTime="600" populationRate="40"/>
    <job type="long" minRunTime="601" maxRunTime="3600" populationRate="30"/>
  </jobs>
  <workload type="unknown" minLoad="20" maxLoad="60"/>
  <termination>
    <condition type="endtime" value="86400"/>
    <condition type="jobcount" value="20"/>
  </termination>
</config>
```




Since the scheduler is only designed to schedule jobs straight away once it receives a new job, there are instances that the algorithm performs worse than the worst first algorithm because the scheduler cannot predict if there will be a job that needs to be performed by the medium server. If the scheduler can obtain all jobs at once, the algorithm will perform better. However, in reality, it is not advisable to wait until every job is known unless the time aspect of the scheduler is not desirable.

# Implementation

## Changes from stage 1

### Client

Previously, due to the simplicity of the ALT algorithm, the client did not utilise two GETs: Avail and Capable. In stage 2, they are used to retrieve the information about the servers. The initial information of the servers is still parsed from the ds-system.xml with ascending order.

### Server Spec

Two new variables are introduced: currentCore and estimatedRunTime. The reason to add them is to have this information available in the ds-client without requesting from the ds-server. They are calculating as followings:

- currentCore is updated every time either GET Avail or GET Capable is called.
- estimatedRunTime is calculated once the server is chosen to run a job:
    - If the server is inactive, estimatedRuntime = boot time + job start time + job estimated run time
    - If the server is running, active or idle, estimatedRuntime = current time + job start time + job estimated run time

### Scheduler

There is a function called sortAsc, which takes care of sorting all the servers in ascending order regarding their cores count.

# Evaluation

The following results are obtained from running the latest test script with provided configurations in the repository:

| **Turnaround time** | ATL | FF | BF | WF | AdvFF |
|---|---|---|---|---|---|
| Average | **254086.33** | 1473.33 | 1462.83 | 6240.72 | **2076.89** |
| Normalised (ATL) | 1.0000 | 0.0058 | 0.0058 | 0.0246 | 0.0082 |
| Normalised (FF) | 172.4568 | 1.0000 | 0.9929 | 4.2358 | 1.4097 |
| Normalised (BF) | 173.6947 | 1.0072 | 1.0000 | 4.2662 | 1.4198 |
| Normalised (WF) | 40.7143 | 0.2361 | 0.2344 | 1.0000 | 0.3328 |
| Normalised (AVG [FF,BF,WF]) | 83.0629 | 0.4816 | 0.4782 | 2.0401 | 0.6790 |

| **Resource utilisation** | ATL | FF | BF | WF | AdvFF |
|---|---|---|---|---|---|
| Average | **100.00** | 66.79 | 64.94 | 72.85 | **82.00** |
| Normalised (ATL) | 1.0000 | 0.6679 | 0.6494 | 0.7285 | 0.8200 |
| Normalised (FF) | 1.4973 | 1.0000 | 0.9724 | 1.0908 | 1.2278 |
| Normalised (BF) | 1.5398 | 1.0284 | 1.0000 | 1.1218 | 1.2627 |
| Normalised (WF) | 1.3726 | 0.9168 | 0.8914 | 1.0000 | 1.1256 |
| Normalised (AVG [FF,BF,WF]) | 1.4664 | 0.9794 | 0.9523 | 1.0683 | 1.2025 |

| **Rental cost** | ATL | FF | BF | WF | AdvFF |
|---|---|---|---|---|---|
| Average | **256.05** | 417.90 | 414.42 | 443.03 | **373.87** |
| Normalised (ATL) | 1.0000 | 1.6321 | 1.6185 | 1.7303 | 1.4602 |
| Normalised (FF) | 0.6127 | 1.0000 | 0.9917 | 1.0601 | 0.8947 |
| Normalised (BF) | 0.6178 | 1.0084 | 1.0000 | 1.0690 | 0.9022 |
| Normalised (WF) | 0.5779 | 0.9433 | 0.9354 | 1.0000 | 0.8439 |
| Normalised (AVG [FF,BF,WF]) | 0.6023 | 0.9830 | 0.9748 | 1.0421 | 0.8795 |

As stated above, the algorithm is inspired by the first-fit, therefore, some of the first-fit algorithm's characteristics are carried over such as low turnaround time, relatively cheap compared to other algorithms (except for ATL).

For the first metric, turnaround time, it outperforms ATL and WF and slightly worse than FF and BF.

As in resource utilisation, it is designed to try to perform concurrencies as much as possible, therefore, it's performance in resource utilisation only comes second after ATL.

In the last objective, there is a significant difference between ATL and AdvFF, however the AdvFF outperforms the FF, BF and WF algorithms.

| | ALT | FF | BF | WF | AdvFF |
|---|---|---|---|---|---|
| Turnaround Time | Poor | Optimal | Optimal | Acceptable | Acceptable |
| Resource Utilisation | Optimal | Poor | Poor | Poor | Almost optimal |
| Rental Cost | Optimal | Poor | Poor | Poor | Acceptable |

## Advantage of AdvFF

- High performance in utilising resources with relatively low price
- The turnaround time is at acceptable range

## Disadvantage of AdvFF

- A speculation is that it will not perform well if the jobs are not in ascending order
- The algorithm is not designed to optimise one object

# Conclusion

The Advance First-Fit algorithm is designed to outperform the best-fit, first-fit and worst-fit algorithms in resources utilisation and rental cost and significantly improved in turnaround time compared to all-to-largest. Even though the AdvFF is performing better than the other algorithms, there is still room for improvement. Through the visualisation of the scheduling, there is an indication that the algorithm can perform better in all three objectives if the scheduler receives all the jobs at once and then performs the scheduling algorithm. Unfortunately, it is not ideal for real life applications since the jobs may be time-sensitive and can cause congestion in the system.

# References

GitHub Repository:
https://github.com/mq-congtrinh-44580460/2021COMP3100Stage2_44580460