Fast Scheduling Algorithm

Name: Hayden Chan

SID: 44817002

1. Introduction

This project will focus on a job scheduling algorithm that is different to baseline algorithms (First Fit, Best Fit and Worst Fit). The goal of the algorithm should implement at least one of the following optimisations:

1. Minimisation of average turnaround time
2. Maximisation of average resource utilisation
3. Minimisation of total server rental cost

2. Problem definition

The performance metrics of this algorithm will be to specifically decrease wait time and decrease execution time which will in turn then reduce the average turnaround time. The baseline algorithms do not achieve this goal except potentially First Fit which could be further improved and optimised.

3. Algorithm description

When investigating the ds-server, ds-client and user-guide given in the ds-sim repository [1], I can start to work out some key performance metrics that will be useful for solving this algorithm problem. When conducting some research I can see that there are the following key values that will need to be considered:

1. End time: The time a job ends
2. Wait time: The total time taken for every job
3. Execution time: The time taken for every job to execute

3.1 Purpose

The purpose of this algorithm implementation is to minimise the average turnaround time. This will be done by looking for the jobs that are the best fit and can be scheduled for any given server, and if this server does not exist then it will simply go onto the next job so we do not waste any time.

3.2 Design

The main factor that makes this algorithm fast is checking if the server had enough resources to do the job at that moment. This causes a change in the outcome as if the servers did not have enough resources to complete the job, it would try again for the next job. This would make it so that jobs cannot be scheduled therefore reducing the average wait time, average turnaround time and simulation end time

4. Implementation

I have created a class for Jobs, this class is based on the specification in the ds-sim user guide and can be used to represent a new Job as an object.
Job objects are created after each incoming message that contains the “JOBN” server command, meaning there is an incoming job.

Data Type	Name	Description
Integer	id	Unique ID of current job
Integer	startTime	The start time of this job
Integer	runTime	Estimated running time for job to complete
Integer	cores	Amount of CPU cores needed
Integer	memory	Amount of memory needed
Integer	disk	Amount of disk space needed

The Server class has a similar structure to the Job class. It also can be used as an object and contains variables based on the ds-sim specification. This is the data types, Names and a short description of each variable:

Data Type	Name	Description
Integer	ID	Unique identifier for each server
String	type	The server type
Integer	state	Current state of server e.g. 0=inactive
Integer	bootTime	Time for this server to start
Integer	coreCount	Amount of CPU cores for this server
Integer	memory	Amount of memory on this server
Integer	disk	Disk space on this server
Integer	waitingTime	Waiting time for for this server
Integer	running Time	Running time for this server

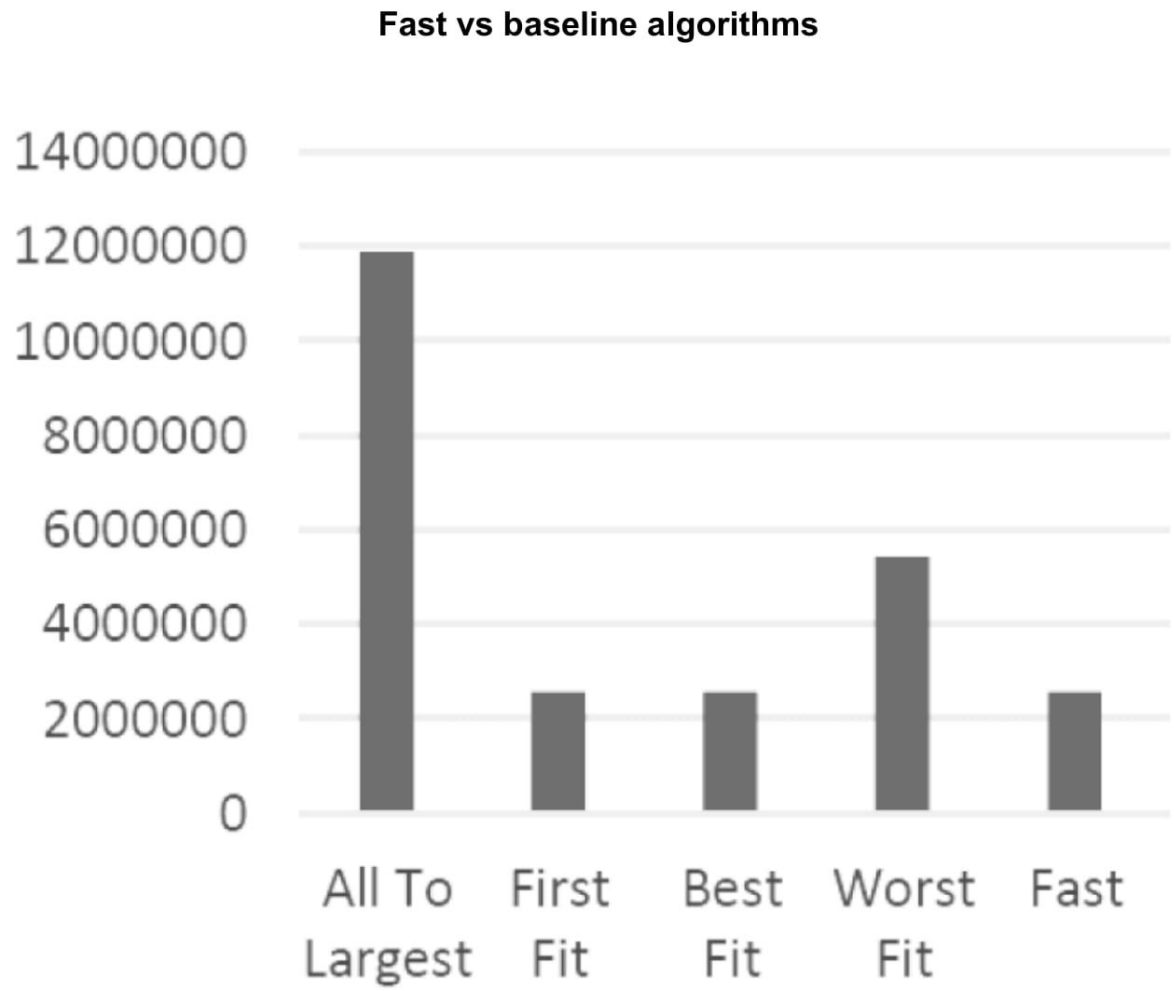
The Server Class has already previously been implemented but just included minor changes however, I had to write a new method to add jobs to a Job ArrayList, this method was called addJob().

5. Evaluation

When evaluating the results of this algorithm I have run it alongside Worst Fit, Best Fit and First Fit then recorded the following results:

	First Fit	**Worst Fit**	**Best Fit**	**Fast**
Simulation End Time	2564544	5437397	2573860	256283
Cost	147135.9	76311.2	115614.4	112315.2
Total Servers Used	60	9	39	17
Wait Time	632	1761109	191172	488
Execution Time	14644	15634	15017	14997
Turn Around Time	15164	1763636	206420	15922

The following bar graph shows a comparison of end times against other algorithms versus the fast algorithm:



From this graph we can see that the simulated times for First Fit and Best Fit are similar to the new fast algorithm however Worst Fit and All To Largest seems to be quite slow. This algorithm shows to be one of the faster ones in this comparison.

6. Conclusion

In conclusion, this algorithm is fast enough and reduces the average turnaround time as well as reducing average wait time. There is no doubt that this algorithm can be somewhat faster given more time and optimisations however this algorithm still beats the other baseline algorithms in testing.

7. References

[1] https://github.com/distsys-MQ/ds-sim
[2] https://github.com/TurboGor/Fast-Scheduling-Algorithm