Project report: Stage 2 - Design and implementation of a job scheduling algorithm

COMP3100 Distributed Systems, S1, 2023

SID: 45353158, Name: Lewis Griffith

Github Repository: https://github.com/Lewis-G/MQU-3100

1 Introduction

The goal of this project [1] was to prompt the student to create a job scheduling algorithm that would be compared against a set of predefined baseline algorithms.

The student's algorithm serves as the client component of the distributed systems simulator, 'ds-sim' [2]. ds-sim replicates job scheduling within computer cluster and cloud environments. In this simulation the client component receives server and job details, and subsequently communicates job scheduling decisions to the server component.

The client component developed by the student will compete against a set of predefined client components that each implement their own algorithms. Each of the job scheduling algorithms are optimized for their own performance metrics and design constraints [3].

Similarly, the student's algorithm was designed with consideration to design constraints as well as the following performance metrics; turnaround time, resource utilization and server rental cost. In developing the algorithm, the student optimized for each of these metrics and learned to weigh these metrics against each other.

Consequently, the goal of this project was for the student to develop an in depth understanding of the following;

- Job scheduling within a distributed system,
- Commonly used scheduling algorithms, and
- The process of developing a new algorithm, including optimizing for performance metrics and other design considerations.

2 Problem Description

2.1 Objective Function

The algorithm should be optimized for the primary performance metric, to achieve a low turnaround time when receiving and scheduling jobs. Additionally, the algorithm should minimize turnaround time without neglecting secondary performance metrics; resource utilization and server rental cost.

2.2 Performance Metrics

The performance metrics are described in detail:

- Turnaround time – Consists of the waiting time plus the execution time.
- Waiting time - Measured as the time taken between submitting a job and that start of the job execution [4].
- Server rental cost – Measured as the resource utilization multiplied by the per-second resource cost. This per-second cost is created by ds-sim.

2.3 Design Constraints

ds-sim outlines a specific protocol by which the client and server components can communicate. [2] The following design constraints are a result of this protocol:

- The client will receive only one job at a time. Job details are sent by the server in the form of a JOBN message after the client sends a REDY message.

- Some jobs may only be runnable on servers that are currently unavailable because they are running other jobs. The client can view a list of available by issuing a 'GETS Avail' request. In the event that this request returns zero available servers, the client can view servers that are capable of running the job later with a 'GETS Capable' request
- To schedule a job the client will issue a SCHD command. This command requires details regarding both the job and the server.

3 Algorithm Description

The student developed two algorithms. Both are variations of the first fit algorithm.

3.1 Stage2 Algorithm

This algorithm consists of a package named, 'Stage2'. It may also be referred to as, 'First Fit with Message Class'.
The basic process of the first fit algorithm is as follows, upon receiving a job:

- The client will request a list of available servers. Upon receiving this list, the client will schedule the job to the first item in this list.
- If there are no currently available servers, the client will request a list of capable servers. Upon receiving this list the client will schedule the job to the first server in this list. In this event, the job joins a local queue for its respective server, and will be run when all of the jobs that are further ahead in the queue have finished.

This first algorithm is optimised using the Message class. This class is used to quickly parse messages sent by the server and obtain job and server information. This class is also used to quickly construct GETS and SCHD commands.

3.2 Stage2Enhanced Algorithm

This algorithm consists of a package named, 'Stage2Enhanced'. It may also be referred to as, 'First Fit with Message Class and Distributed Local Queues'.

This algorithm builds upon the ideas of the prior algorithm. The differences between the two algorithms occur in two places.

Firstly, the client will issue a GETS ALL request at the beginning of its execution. The request will return a list of every server. The client will store details of each server in a Server class. This class will capture the server IDs, server type, and the number of queued jobs belonging to the server. At the start of the program execution, each server, will contain zero queued jobs.

The second difference will occur when the client receives an empty list of available servers. Upon receiving this empty list, the client will request a list of capable servers. The client will search through this list for the server with the fewest queued jobs. Upon selecting a server, the client will locate the respective server details within the server class, and increase the number of queued jobs.

This optimisation ensures that jobs are not being queued to the same capable server several times. Instead the jobs will be distributed across many different servers.

3.3 Example of Algorithm

An example of the algorithm execution is laid out in figure 1. This example depicts 6 jobs being scheduled amongst 3 servers. The server names are 'Small', 'Medium', and 'Large'. The number of rectangles within each server depicts their number of cores.

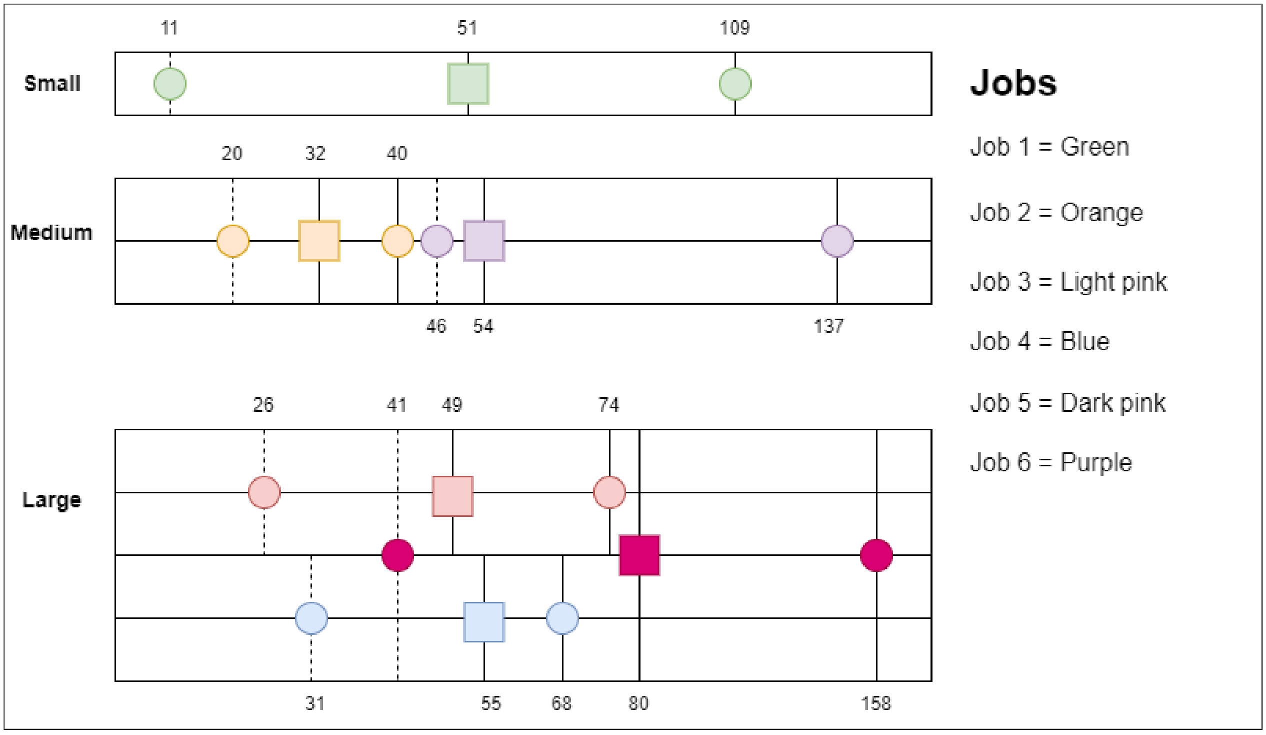
Each of the jobs are colour coded. Each job consists of three events;

- Job submission - The client receives the job details from the server and sends a SCHD command. This is shown with a circle and a dotted line.

- Job start - After the job is submitted, their will be a waiting time until the job begins running on the server. In the event that a job is queued to a server, there will be an extended waiting time. This is shown with a square and a solid line.
- Job completion - The job has been completed. The next job in the queue may now begin. This is shown with a circle and a solid line.

This example is hypothetical. The time for each event is given in seconds.

Figure 1: Example of algorithm execution



3.4 Discussion of Example

The example provided above can be used to demonstrate either of the students algorithms. These algorithms will iterate through the jobs, scheduling jobs to the first available server. By job 5, there are no avaliable servers. As such, the job is scheduled to the only capable server, the large server.

Whilst job 5 is waiting to start, the client schedules one more job. By 158 seconds, all jobs have been completed.

4 Implementation Details

The implementation can divided by the java Classes. Both algorithms utilize the 'MyClient' and 'Message' classes. Stage 2 enhanced implements both of these classes, with modifications, as well as an additional 'Server' class.

4.1 MyClient Class

This class serves as the main body of the algorithm. It contains attributes of the following types; a 'Socket', 'BufferedReader', and 'DataOtputStream'. These attributes are used for communication with the server component of ds-sim. It also contains a Message attribute. The enhanced algorithm contains a Server attribute. It contains the following methods:

- Constructor - Upon calling this class a socket connection is established with the server component of ds-sim.

- send() - Sends messages to the server.
- receive() - Read the message from the server.
- receiveString() - Read the message from the server and return the result as a string.
- authenticate() - Complete the authentication process, laid out by the ds-sim protocol.
- receiveGets() - Sort through the response to the GETS command. Save the server details of the first server and then schedule a job using the Message attribute.
- endConnection() - Close the socket connection.

4.2 Data Structure (Message Class)

This class holds job and server details with the class attributes. Most of this attributes are self explanatory:

- An integer, 'jobID'
- An integer, 'jobCores'
- An integer, 'jobMemory'
- An integer, 'jobDisk'
- A string 'serverType'
- An integer, 'serverID'
- An integer, 'nRecs'. This attribute holds the number of records given after issuing a GETS request.

4.3 Data Structure (Server Class)

This class consists of two attributes, both are array lists.

'serverTypes', is an array list holding String objects.

'queueLengths' is an array list holding integer arrays. The integer arrays correspond to the number of queued jobs.

The indexes of both array lists correspond with each other. For example, to find the queue length of the tiny 0 (server type is 'tiny' and server ID is '0') you must get tiny servers index from 'serverTypes' and then return the first item in the corresponding array list, located within 'queueLengths'. Together these attributes provide a means of quickly identifying the queue length of a respective server.

5 Evaluation

5.1 Test Cases

These algorithms were test using the test suite provided in week 11 workshop. As seen below, each algorithm was run against each other, with their turnaround time, resource utilisation, and total rental cost being recorded.

Figure 2: Averaged test suite performance metrics

Metric	Stage2	Stage2Enhanced	FF	BF	FFQ	BFQ	WF
Turnaround time	1342.4	1253.2	1867.13	2417.27	2559.07	2401.80	9958.4
Resource utilisation	73.63	72.3	74.02	69.55	73.49	63.95	68.18
Total rental cost	554.15	587.99	526.40	524.82	537.31	531.21	555.13

5.2 Algorithms Pros and Cons

Both Stage2 and Stage2 enhanced offer considerable improvements in turnaround time, in comparison to the baseline algorithms. Stage2Enhanced in particular offered a very low turnaround time.

Both algorithms provided slight improvements in resource utilisation, in comparison to the averaged baseline algorithms.

Both algorithms, suffered in terms of total rent cost, in comparison to the base line algorithms. This likely because both algorithms have a tendency to allocate jobs to larger servers.

Given that the primary performance metric was turnaround time, the losses in total rental cost, were judged to be acceptable.

6 Conclusion

Ultimately, the student developed two separate algorithms. Each algorithm had its own strengths and flaws. The Stage2 algorithm performed well in all metrics, where as the stage2Enhanced algorithm performed even better in turnaround time, but poorly in total rental cost.

Through out this assignment, the student gained an understanding in the following areas:

- Job scheduling within a distributed system,
- Commonly used scheduling algorithms, and
- The process of developing a new algorithm, including optimizing for performance metrics and other design considerations.

References

[1] L. Griffith, “MQU-3100,” Apr. 2023.

[2] L. Young Choon, K. Young Ki, and J. King, *ds-sim: an open-source and language-independent distributed systems simulator*. Macquarie University.

[3] Y. P. Dave, A. S. Shelat, D. S. Patel, and R. H. Jhaveri, “Various job scheduling algorithms in cloud computing: A survey,” in *International Conference on Information Communication and Embedded Systems (ICICES2014)*, pp. 1–5, 2014.

[4] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating system concepts, Chapter 5*. Wiley, 2010.