Project report: Stage 2
COMP3100 Distributed Systems, S1, 2022
SID: 46133194, Name: Man Chun Kwok

1 Introduction

This report is to introduce the COMP3100 Assignment Stage 2 project. Stage 2 is design and implement an algorithm that optimises average turnaround time. Client simulator should schedule the jobs to server and minimise the average turnaround time.

To schedule the jobs in this stage, it should design an algorithm is minimising the average turnaround time, but the cost and resource utilisation needs to be within a reasonable range.

In this report, I aim to introduce how to design and implement an algorithm for schedule the jobs. The rest of this report is organised as follows.

- Section 2 gives the problem definition.
- Section 3 gives the algorithm description.
- Section 4 gives the implementation details of algorithm and including the data structures used.
- Section 5 gives the evaluation of the algorithm.
- Section 6 will summarize the finding while implementing the algorithm.

2 Problem Definition

There are many server types. Because of the turnaround time, cost and resource utilization, it is very difficult to choose only one of the most suitable server types. In order to select the appropriate server type, three goals need to be set. This includes reduced turnaround time, reduced cost and reduced resource utilization.

Start from first capable, because it is a basic schedule algorithm. It schedules a job to the first server which response from server.

In order to achieve the above three goals, the following considerations are therefore taken.

1. To reduce turnaround time, only increase the resource utilization to each server type but not too much.
2. To reduce resource utilization, only prioritize just enough resources to perform the current job.
3. To reduce the cost, only use server types with lower core counts as much as possible.

3 Algorithm Description

In order to achieve the above three condition, there are the following methods.

1. Find the smallest fitness value from server types list.
2. Find the largest fitness value from server types list.
3. Each server type should be used at least once and reuse the server type when it active and do not has a job scheduling.

To identify the best server type, the list should be use to filter the server type which is sufficient resource capacity regardless of availability by using smallest **fitness value** and not having any jobs and waiting jobs at the same time because it should be confirm the job can be running as soon as possible to complete the first two goals. The **fitness value** is calculate by the available cores of server type minus the core requirement of the job. If there are more than one fitness value is the same, select the first one according by initial core count.

Moreover, the server types may having any jobs and waiting jobs at the same time. It should be identify the best server type by another filtering. The condition should be only get the smallest fitness value to confirm the cost is low.

Finally, the server types may not have the fitness value larger than or equal to 0. It should identify the best server type from the list which filter condition should be changed to get the largest fitness value to confirm the next job of fitness value scheduling will not be affected.

For example the server type are:

Server Type	Core	Fitness Value
small	2	-
medium	4	-
large	5	-

Table 1: Server types sample.

When receive three jobs all needs 2 cores:

Server Type	Core	First Fitness Value	Second Fitness Value	Third Fitness Value
small	2	0	-2	-2
medium	4	2	2	0
large	5	3	3	3

Table 2: Server types sample with 3 jobs of 2 cores.

According to table 2, the job has been scheduled to small first because the fitness is larger then or equal to 0. Then the job scheduled to medium because the positive smallest fitness value is medium server type. Finally, we can see that the last job will make medium fitness value to 0 which match the condition. The job will be scheduled to medium server type.

When receive three jobs all needs 4 cores:

Server Type	Core	First Fitness Value	Second Fitness Value	Third Fitness Value
small	2	-3	-3	-3
medium	4	0	-4	-4
large	5	1	1	-3

Table 3: Server types sample with 3 jobs of 4 cores.

According to table 3, the job has been scheduled to medium because the fitness is smallest and larger then or equal to 0. Then, schedule to large. After that, all fitness values are the negative. There are only large server type can take the job in future because medium server type may done a job and get the fitness value 0 easier than large server type.

4 Implementation

There are only use a functions in the code which is called **getNextServerTypeAndIdFromRecord** [1]. The algorithm is using **Stream** and **Lambda** to complete.

To use **Stream** and **Lambda** to complete the algorithm, it should design the data structure.

- Data Structure Class
  - The responsibility of the data structure classes are to record the data. When receive the data from the server, data structure classes should record the data in the memory. **JOBNcmd** class record the important data from JOBN request. **ServerType** class record the list of server types data.
- Listing
  - The responsibility of the list is to record all the server types which receive from server. **list** recorded all the server types and It will be used by **Stream** and **Lambda**.

The following is the pseudo code of the algorithm

```
select all the server types
  which fitness value larger than or equals to 0
  and not having any running and scheduled
and get a server type which is minimum fitness value in selection

if not found
  select all the server type
    which fitness value is larger than and equal to 0 and active
  and get the minimum fitness value in selection

if not found
  select the largest fitness value in list
```

5 Evaluation

According to table 3, table 4 is the continue. When receive a job needs 4 cores.

Server Type	Core	Third Fitness Value
small	2	-3
medium	4	-4
large	5	-3

Table 4: Server types sample continue table 3.

According to table 4, this is the only different to the FF, BF and WF algorithm. FF, BF and WF algorithm will select the medium because the medium initial cores is enough and it is the smallest cores of server type which is available to schedule. However, my algorithm is select the large one because it should fit for the future.

Select the largest fitness value may add a little bit cost because more cores more cost. When the future jobs cores is a few, it is a good choice because not all needs require a large number of cores. Even if a large number of cores are required, the turnaround time is guaranteed to a certain extent.

The only pros of the algorithm is the turnaround time is better than FF and BF. Because FF and BF did not give much thought to the unknown and just schedule the work in order. The cost and resource utilization may be the same. There is not much difference between them. It seems my algorithm is better.

6 Conclusion

In conclusion, implementations of this algorithm is a few but complex. There are multiple situations to consider. This algorithm is like a collection of BF and WF. If the premise of BF is not satisfied, the way of thinking of WF is used to operate.

In this assignment, I found that the algorithm is important. It represents whether the operation of a distributed system can dynamically allocate a variety of general physical and logical resources.

References

[1] M. C. Kwok, “Comp3100 assigment.” https://github.com/8593K/COMP3100, 2022.