Cloud Job Scheduler & Discrete Event Simulation

Name: Dac Minh Quang Nguyen

StudentID: 45506698

Introduction

This project is to introduce the basic concept of a distributed system, also known as distributed computing. This system has multiple components placed in several machines such as computers or any devices that have local memory and can process multitasking. Those components communicate and interact with each other via a network to achieve designed tasks. A Distributed System is also able to scale up to millions of servers to meet a demand on the explosion of technology nowadays such as Google Data Centres, AWS clouds.

In the approach to the Distributed System, in the Previous task, we designed a simple system that can dispatch jobs to an appropriate server which, in this case, is the biggest possible server. In Stage 2 of the process, instead of assigning to the biggest sever, we are not applying some algorithms called First Fit (ff), Best Fit (bf), and Worst Fit (wf) to have better performance for some purposes which are Turnaround Time, Rental Cost, and resource utilization. This report will take into account the logic and evaluation of each base-line algorithm, implementation details

Problem Define

In the Previous Stage, when we assign all job to the largest server, it is clear to the observer that the turnover time consumes way too much while others do not get that perform. In detail, the performance of turnaround time takes over approximately 20000% worse than a first fit baseline algorithm, while cost and resource utilization have improved only about 25% each. This trade should not have happened, therefore for better performance, applying those algorithms will distribute all resources reasonably. After

applying those algorithms to this project, turnaround time, resource utilization and cost are well-sense now, with huge improvement on turnaround time, the resource utilization and cost are only dropped by about 15%.

The logic of Each Algorithm

First Fit Algorithm:

In the context of OS memory allocation, it assigns data into the memory by scanning from the beginning of available memory to the end, until the first free and appropriate space is found.

In the context of job scheduling, instead of searching for available memory, we search for the first available core count and having no waiting job. If no server is found, then the first available server will be assigned.

For example:

GETS CAPABLE 2 2000 4000 ->

Abcd 0 inactive -1 8 16000 52000 0 0

Jinn 0 inactive -1 4 16000 52000 0 0

Joon 0 inactive -1 16 16000 52000 0 0

We will select server **Abcd** since it comes first.

Best Fit Algorithm

In the context of OS memory allocation, it fits data into the smallest available partition of memory.

In the context of Job Scheduling, instead of searching for available memory, we search for the smallest core count and having no waiting job, if not server is found, then first sorted available server will be assigned.

We will select the **Jinn** Server since it is the smallest and most appropriate server.

Worst Fit Algorithm

In the context of OS memory allocation, it fits data into the largest available partition of memory.

In the context of Job Scheduling, instead of searching for available memory, we search for the largest core count and having no waiting job, if not server is found, then the first sorted available server will be assigned.

We will select the **Joon** Server since it is the largest and most appropriate server.

Implementation

For implementation, the greeting process is staying the same as in Stage1, input, output data streams and other utility libraries need to be imported for the connection and working algorithm. First, the Client has established a connection using localhost and port 5000. Second, send “HELO” to confirm and send “AUTH” to confirm the user to who the server talking. After it, everything will be in a loop until no more job needed to assign (“NONE”). If “JOBN” sends, then it is time to start on the Algorithm.

First, we take the argument terminal in terms of ff, bf, wf, and implement it based on each algorithm logic. The sort function is taking into account since it is needed to sort particularly for Best Fit and Worst Fit.

Evaluation

Every algorithm has its strength and weakness, and its strengths also depend on the size of JOBs that are needed to assign. We have three mainly measuring components which are Turnaround Time, Resource Utilization and Rental Cost. We are unable to get all best of the three without any sacrifice. The following will analyze each method and give actual results.

First Fit Evaluation

Advantages

- The implementation for First Fit is simple.
- It takes a shorter time (running time) since no sorting has been used.

Disadvantages

- Get worst when jobs and servers have diverse sizes
- left unused with later memory since its order at last

- no stability and easily **Fragmentation**.

I have tested with many test cases and observed that, if a server is already sorted then it performs very well (even better than best fit because the logic of the first fit is similar as best fit but sort part). However, when it comes unsorted, the story comes way too different, it is like a lucky wheel, some are bad, some seems good in terms of Turnaround Time, Resource Utilization and Rental Cost.

Best Fit Evaluation

Advantages

- Memory Efficiency
- Optimize everything (eg. Total Cost, turnaround time)
- Performs well when all jobs and servers are in relative sizes.
- It is simple enough to implement

Disadvantages

- It takes a longer time (running time) since sorting has been used

I think this is the best way in this project all over two other algorithms because job sizes and server sizes are similar (not like very big and very small). I have tested our project with many test cases, and observed that this logic has optimized the Turnaround Time, and Rental Cost but Resources Utilization,

Worst Fit Evaluation

Advantages

- Performs well when all jobs and servers are in **different** sizes, huge sizes.

Disadvantages

- It takes a longer time (running time) since sorting has been used

I have tested it with many test cases it performs very well in Resources Utilization and Turnaround Time.

Combining Methods

The best solution is to combine those 3 methods, in details, when the server is already sorted then the system should work on First-Fit. If it is not sorted and sizes are relative, then Best Fit would be the first choice. If it is not sorted and sizes are massively different, Worst Fit would be the best choice. The combination of three will bring the best method, unfortunately, I have failed to implement it to give an actual result, and observation of how well it is.

Conclusion

Throughout Stage 1 and Stage 2, I have had a brief understanding of how Distributed System working by sending and distribute information to the Server. More importantly, I have learned three types of the algorithm (FF, BF, WF) that are currently in use (for example in memory allocation). Each has its own advantages and disadvantages, in particular:

First Fit: Faster than others when allocating to memory (no sort is needed) but it tends to waste the later memory since its order is at last, and has a large turnaround time.

Best Fit: it has better memory efficiency obviously and turnaround time, but slower process compared with First Fit and worst performance in resource utilization.

Worst Fit: it has a better performance in resource utilization since everything has put in the largest first, however, it has slower processing time compared with both FF and BF.

REFERENCES

https://www.javatpoint.com/socket-programming

https://www.geeksforgeeks.org/socket-in-computer-network/

Project Git Repository

https://github.com/DacNguyen234/Comp3100_project