Resource Utilization and Cost Reduction in Distributed Systems

Andrew Campbell – 45638292

Introduction

The aim of this project is to create a client for the ds-sim server that takes a job and schedules it to a server based on different factors that will allow it to reduce the overall cost and improve the resource utilization of the servers being used.

This stage of the project focuses on taking one of the pre-defined algorithms (first-fit, best-fit or worst-fit) and improving on it to achieve at least one of the following:

* Minimisation of average turnaround time
* Maximisation of average resource utilisation
* Minimisation of total server rental cost

In this instance, the best-fit algorithm is being improved upon to minimize server rental cost and maximise average resource utilisation.

Problem Definition

The best-fit algorithm makes use of the fitness value calculated using the CPU cores required vs the CPU cores available from a server. Although this method is effective in finding a server that would leave the least amount of CPU cores unused, when choosing a server, it does not take into account the ability to run multiple jobs simultaneously. To improve on the best-fit algorithm, my objective function will look at not only the fitness value of the available servers, but will also look to see if any current jobs are being completed by those servers, giving higher priority to servers that already have jobs running that are capable of running the current job simultaneously.

Algorithm Description

Implementation details

The program has been split into five main classes that hold their own functions. The classes are split up as per the following; client.java, bestFitFinder.java, handshaker.java, msgReader.java, msgWriter.java. The main function is implemented in the “client.java” file and is used to call functions from other classes to allow interaction and scheduling with the server. The main function makes use of the handshake function found in the handshaker class to make the handshake with the server at the start of the interaction between the client and the server. The main method then uses the msgRead and msgSend functions found in the msgReader and msgWriter classes respectively to talk to the server, depending on what the server has sent through. The main function requests a job then requests the available servers and chooses the best server out of the available servers by passing the CPU Cores required and the available servers to the bestFit function in the bestFitFinder class. The bestFit function compares the available servers requests to see their waiting jobs from the server by using the msgRead and msgSend functions. The bestFit function then returns the best server to be scheduled by the main function.

Evaluation

Conclusion

In conclusion, whilst working towards a solution to maximize the resource utilization of my client when compared to the Best Fit algorithm, I was also able to minimize the running cost of the servers. This was able to be done by the implementation of my client of which its main priority was to make as much use of all CPU Cores available in any given server at any given time.

References