**COMP3100 – Distributed Systems Task 2 – Improved Best Fit**

**Introduction**

Scheduling algorithms are essential for distributed systems. They allow us to allocate resources efficiently and prevent waste of resources. There are many different types of scheduling algorithms. Three well-knows algorithms include Best-Fit, Worst-Fit, and First-Fit. Each with their own distinct advantages and disadvantages. These three algorithms have been explored throughout the semester of COMP3100, however these are not the only approaches to the scheduling of tasks.

**Problem Definition**

A solution to provide a more efficient approach to the scheduling of tasks between a server and client.

**Algorithm Description**

Task 2 sees an opportunity to experiment and test the boundaries of scheduling algorithms. The chosen approach sees the consideration of all the servers in a simulation and determining which server is the best to schedule the incoming job.

The selected algorithm prioritises a fitness value-based system, where the required cores of the job is compared to the cores of each server. The algorithm will then select a server that currently does not have a waiting job.

Consider, the AllToLargest algorithm, we see that all jobs are scheduled to the largest available server, without consideration of external metrics or factors. This is a poor utilization of available resources as the servers in our control are not being used. This also sees a huge turnaround time and is not favorable in a real-world scenario.

In this way, instead of instantly scheduling the incoming job, as in stage 1, we carefully consider each server and select it, if it fits our specifications.

**Implementation**

Implementing this algorithm saw the employment of GETS Capable and adding each of the servers that can support the incoming job to a list. Once added, we can parse each server to and arraylist which will allow us access metrics such as current start time, core count, memory, disk, waiting jobs and running jobs. These metrics can help to determine whether a server is suitable to schedule the task to. Once parsed, we can begin to traverse through all capable servers and obtain a fitness value. With this fitness value, we can determine whether a server is suitable and schedule it accordingly.

**Evaluation**

This algorithm while simple, does at times outperform Best-Fit in terms of cost effectiveness with a higher percentage of resource utilization. On some configs run with the test script, we saw a increase in the cost effectiveness, an improvement in resource utilization and rental cost, however on some config files, we also saw a decrease in performance across the board. The strongest negative of the algorithm is its little deviation from the best fit algorithm.