Group Members

- Ang Ze Yu (A0187094U)

- Choo Xing Yu ()

**Program Design**

We will explain our program design from the 0% to 100% case (not hardcoded), showing how the addition of job knowledge influences our scheduling algorithm.

0% case

For the 0% case, we have no knowledge on the jobs and their sizes. Therefore, even though we can measure the elapsed time of each job, we cannot make any assumptions about the server capacity since we do not know the job size.

Therefore, our design here centers around a simple principle:

*Assign whatever work there is to whatever resource is idle / available*

Specifically, we:

* limit the number of jobs processed by each server to **1** at any time
* when a new job request comes in, we iterate through the servers and find **any first server that is idle**. We assign the job to this server.

In doing so, we **exploit the uncertainty and unevenness of server processing capacities**. This is because **faster servers** will **process jobs faster.** This leads to them becoming **“idle” faster, and more times than other servers**, which our algorithm naturally takes advantage of.

50 & 100% case

Here, we have knowledge on job sizes. We can therefore **estimate the processing capacities of servers** by  **= job size / time elapsed**.

So, for the first **N (number of servers)** job requests, we assign the requests to servers whose capacities **are still unknown**.

Once that is done, we exploit our server capacity estimations in the following way for each job (**whose size is known**):

1. For each server, we find a **response\_time** estimate by
   1. Calculating the **process\_time\_needed** for the current job (job size / capacity)
   2. Calculate the **queue\_time** resulting from jobs being processed (if there is one) by using the time elapsed since the previous job’s start and process time of the previous job
   3. Sum these together
2. We choose the server with the best response\_time and assign it

For jobs **whose size is unknown (50% case)**,