**ASA Distributed Systems Job Scheduler**

**Ahmad Sohial Ahmadiar (45129223)**

1. **Introduction**

In stage 2, the goal is to extend the client that was implemented in stage 1 through implementing a new algorithm that improves on the baseline algorithms given of Best-fit, First-fit or Worst-fit.

The new algorithm should improve on either the:

* turnaround time,
* resource utilisation, and/or
* server rental cost

of the 3 mentioned baseline algorithms provided. By trying to improve on one of these there will likely lead to the worsening of another for example improving the turnaround time it will likely need to greater server rental cost. Thus, as the programmer, I must decide on whether it is worth the trade off or not, by seeing what this will be used for. In the real world this would be discussed with all involved in the project.

1. **Problem Definition**

The problem of the scheduling is that it is not possible to improve on all three metrics of

* turnaround time,
* resource utilisation, and/or
* server rental cost

therefore, improving on one metric will likely worsen the others.

With this in mind, one must decide on which to improve and which to accept that will worsen. Hence, I had chosen to focus on improving the Best-fit.

1. **Algorithm Description**

With trying to improve on the best-fit model, I needed to bring down the turnaround time by sending a job with next available server. For example, if a job needs 1 core and the other servers are occupied but the server with 4 cores has 1 core available, we send the job to that server to run concurrently with the old job.

1. **Implementation**

The program imports several libraries, essential to its successful running:

* java.io.\*
  + Java IO is the library that allows programmers to utilise basic input and output. This is especially important for our program given our requirement to have constant communication with other programs i.e. ds-server.
* java.net.\*
  + Java.net is responsible for allowing programmers to utilise structures such as Sockets, which are evidently crucial to this project. It is the way in which we initialize the server socket and ultimately communicate.
* java.util.\*
  + Java.util is crucial for allowing the program to use ArrayLists, which let the programmer import the Server structure into an Array.
* javax.xml.parsers.\*
  + This is a self-explanatory library that allowed us to parse the XML, a crucial step in understanding the server properties.
* java.org.w3c.dom.\*
  + This library allowed us to use a NodeList which assisted in the parsing of the XML and allowed us to hold variables in perpetuity within the List.

Due to my Algorithm not being able to be run, I can only speculate of the pros and cons of it.

* 1. **Pros**

Best for projects that are time sensitive.

* 1. **Cons**

The cost would significantly increase.

1. **Conclusion**

In real life an improved best-fit algorithm would be ideal if cost was no issue in the project, therefore it would only be ideal in time sensitive applications.

1. **References**

Github:

https://github.com/sohero31/COMP3100-client