Stage 1: Design and implementation of a client-side simulator with a simple job dispatcher
DUE 5pm Monday, Week 7

This assessment item accounts for 20% of the total marks (100) of the unit. This simulator design and implementation is set to be **individual** work. The marking will be conducted in two steps: (1) assessing the working of your stage 1 code in a demo session (at your allocated workshop in Week 7), and (2) offline marking on the report, project management and the code.

Your task in this stage is to design and implement a plain/vanilla version of client-side simulator that acts as a simple job dispatcher. This version of client-side simulator shall implement:

- The ds-sim simulation protocol as described in the ds-sim user guide. Briefly, it connects the server-side simulator (ds-server), receives jobs (one job at a time) and schedules them.
- A simple job dispatcher/scheduler, called Largest-Round-Robin (LRR) that sends each job to a server of the largest type in a round-robin fashion. The largest server type is defined to be the one with the largest number of CPU cores. If there are more than two server types with the same number of cores, use the first one, i.e., the first one listed in the configuration file or ds-system.xml. See the example below:

  For the largest server type of *xlarge* with five servers, LRR schedules

  job 0 to xlarge 0, job 1 to xlarge 1, ..., job 4 to xlarge 4, job 5 to xlarge 0, job 6 to xlarge 1, ...

  You can assume there is no job that requires more resources than those of a largest server; this has been ensured in the current implementation of ds-server job generation.

The reference implementation of client-side simulator (`ds-client`) can be found in the ds-sim repository (https://github.com/distsys-MQ/ds-sim). Your implementation of ds-client is required to work the same way as the reference implementation. In particular, for a given simulation with the '-v brief' option[1] for ds-server, your simulation log should match exactly that generated by the reference implementation.

Your submission for this stage is ONLY your **5-page** report that must contain the link to your project git repository (e.g., GitHub and Bitbucket). You are required to use LaTeX (e.g., Overleaf) for your report. The LaTeX template for your report will be provided. You submit only the PDF file of your report. Make sure your project git repository is accessible by the markers (COMP3100/6105 teaching staff). In other words, you may either make it public or explicitly give access to them.

Your demo must be done with the version of your client-side simulator at the time of submission (NO changes allowed after the submission). Any necessary files, such as *makefile* or a shell script for installation and a user guide should be available in your git repository for the markers to reproduce what you show in the demo session. More detailed instructions, such as configuration files and the script to auto-run your demo for your demo will be given before your demo. Note that your client-side simulator should work with any simulation configuration.

"Suggested" headings of stage 1 report are as follows (STRICTLY 5 pages max including everything):

1. Introduction (1/2 page[2]): What this project (focusing on Stage 1) is about, including the goal of the project and Stage 1.
2. System overview (1/2 page): high-level description of the system (both client-side simulator and server-side simulator with the focus being your client-side simulator), preferably, with a figure (your own, not one in the ds-sim user guide) showing the workflow/working of the system
3. Design (1 page): design philosophy, considerations and constraints, functionalities of each simulator component focusing on the client-side simulator
4. Implementation (2 pages): brief description of any implementation specific information including technologies, techniques, software libraries and data structures used. How each of components/functions of your simulator is implemented
5. References including your project git repository link, e.g., GitHub and Bitbucket

If you would like to add a figure other than your own (e.g., a workflow of discrete event simulation from a research paper), you should either get a written permission from the author(s) or redraw it yourself and cite the source.

[1] If you run the server with the '-v all' option, ds-server might show the log differently depending on how the client makes scheduling decisions.

[2] The number of pages for each section is only indicative.

Marking rubric (in percentage)

85 to 100

Work in this band presents a full and comprehensive account of all requirements outlined above. In particular, the efficient and elegant design and implementation of fully functional client-side simulator with the correctly working LRR algorithm should be evident in (1) the source code, (2) the report and (3) the demo. The clear demonstration of good project management should be evident in all these three components, including the git commit history (constant/regular commits throughout the duration of project).

75 to 84

Work in this band presents a clear account of all requirements outlined above. In particular, the good design and implementation of fully functional client-side simulator with the correctly working LRR algorithm should be evident in (1) the source code, (2) the report and (3) the demo. The sufficient demonstration of good project management should be evident in all these three components, including the git commit history throughout the duration of project.

65 to 74

Work in this band presents an adequate design and implementation of working client-side simulator. The appropriate demonstration of good project management in all three assessment components should be evident.

50 to 64

Work in this band presents a decent design and implementation of working client-side simulator. The appropriate demonstration of project management should be evident.