Stage 2: Design and implementation of a new scheduling algorithm
DUE 11:59pm Monday, Week 13

This assessment item accounts for 20% of the total mark (100) of the unit. It is set to be individual work. Similar to Stage 1, the marking will be conducted in two steps: (1) assessing the working of code at the demo (at your allocated workshop in Week 13), and (2) offline marking on the report and code[1].
By now, you should be well aware how jobs are dispatched/scheduled in distributed systems, like data centres simulated by ds-sim. In such scheduling, there are a number of performance metrics/objectives and constraints, such as execution time, turnaround time, resource utilisation and costs of execution (to be discussed in the next few lectures).

Your task in this stage is to design and implement one or more new scheduling algorithms that optimises average turnaround time. In particular, your scheduling algorithm as part of your client shall schedule jobs to servers aiming to minimise the average turnaround time without sacrificing too much of other performance metrics, i.e., resource utilisation and server rental cost.

As these objectives are incompatible (conflicting performance objectives), the optimisation of one objective might lead to sacrificing the optimisation of other metrics. For instance, if you attempt to minimise average turnaround time by minimising waiting time, you may end up using more resources resulting in high costs (i.e., server rental costs). Therefore, you have to define the scheduling problem clearly indicating your performance objective(s) and justifying your choice in the report. In addition, you must discuss your algorithm and scheduling results in comparison with baseline algorithms (FF, BF, FFQ, BFQ and WFQ) and their results.

As in Stage 1, your client should work for any simulation configuration; that is, you shouldn't assume those sample configurations given before are the only ones used for testing and marking. You may even create your own configurations to more effectively demonstrate the performance of your algorithm; these configurations, if used in your discussion (report), must be included in your submission.

The submission shall be ONLY your **5-page** report that must contain the link to your private project git repository (e.g., GitHub or Bitbucket). You have 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 marker (your tutor). In other words, you should make it private and share it with your tutor.

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 will be given beforehand your demo. Note that your client-side simulator should work with any simulation configuration.

"Suggested" headings of Stage 2 report are as follows. Your report should be no more than 5 pages including references and appendices. Note that the number of pages for each section specified below is only indicative.

1. Introduction (1/2 page): What this project (focusing on Stage 2) is about, including the goal of the project and Stage 2.
2. Problem definition (1/2 page): the description of the scheduling problem and the definition of your objective function including the justification of your choice.
3. Algorithm description (1 page; one sub-section per algorithm if you have more than one); **NB: You need to provide a simple example scheduling scenario, including a sample configuration, the schedule, the description and discussion**; this is to 'visualise' how your scheduling algorithm works.
4. Implementation details including data structure(s) used (1/2 page).
5. Evaluation (2 pages): simulation setup including test cases/configurations, results, comparisons with baseline algorithms and discussion including pros and cons of your algorithm. Use figures and tables effectively and be space conscious.
6. Conclusion (1/4 page): summary + what you have found and what you suggest.
7. References (1/4 page) including your private project git repository link, e.g., GitHub or Bitbucket

[1] We will use a code plagiarism check tool as in Stage 1.

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 one or more fully functional and robust new scheduling algorithms should be evident in (1) the source code, (2) documentation and (3) demo. One algorithm at least should demonstrate superiority over baseline algorithms in its design and performance in terms of all performance metrics. The performance of scheduling algorithm(s) needs to be extensively and thoroughly analysed and justified both qualitatively and quantitatively in appropriate forms, such as tables and comparison charts. The solid understanding of Stage 2 should be evident in the submission and demo, and the project management, e.g., git commit history.

75 to 84

Work in this band presents a clear account of all requirements outlined above. In particular, the efficient and clear design and implementation of one or more fully functional (robust) new scheduling algorithms should be evident in (1) the source code, (2) documentation and (3) demo. One algorithm at least should be sufficiently different from baseline algorithms with its performance equivalent to or better than that of baseline algorithms based on one or more performance metrics without sacrificing any performance metric too much compared to the performance gain for the defined objective. The performance of scheduling algorithm(s) needs to be analysed and justified both qualitatively and quantitively in appropriate forms, such as tables and comparison charts. The solid understanding of Stage 2 should be evident in the submission and demo, and the project management, e.g., git commit history.

65 to 74

Work in this band presents the good design and implementation of one or more new scheduling algorithms with clear algorithm description and discussion. One algorithm at least should demonstrate some performance advantage over one or more baseline algorithms. The performance of scheduling algorithm(s) needs to be analysed and justified both qualitatively and quantitively in appropriate forms, such as tables and comparison charts. The clear understanding of Stage 2 should be evident in both the submission and demo.

50 to 64

Work in this band presents the adequate design and implementation of at least one new scheduling algorithm with appropriate algorithm description. Appropriate discussions are required in documentation with identification and justification of performance of the scheduling algorithm. The good understanding of Stage 2 should be evident in both the submission and demo.