**Stage 2: Design and implementation of a new scheduling algorithm (DUE 5pm Monday, Week 13)**

This assessment item accounts for 20**%** of the total mark (100) of the unit. It is an individual work; hence, individual submissions. Similar to Stage 1 marking, 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.

By now, you should be well aware how jobs are dispatched/scheduled in distributed systems, like data centres simulated by ds-sim, works. 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, Week 9's in particular).

**Your task in this stage is to design and implement at least one new scheduling algorithm that optimises one or more objectives.** In particular, your scheduling algorithm as part of your client shall schedule jobs to servers aiming to achieve one or more of the following:

- Minimisation of average turnaround time
- Maximisation of average resource utilisation
- Minimisation of total 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 addition, you must **discuss** your algorithm and scheduling results **in comparison with three baseline algorithms (FF, BF and WF)** and their results.

As in Stage 1, your client should work for any simulation configuration; do not 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.

**Deliverables**

The submission is to be **a single compressed file** of your project git repository ('Download ZIP' in GitHub and 'Download repository' in Bitbucket). You may 'fork' Stage 1 git repository or create a new one.

Your submission shall contain at least:

- The **source code** of your scheduling algorithm as part of your client*
- The **report**
- *Custom configuration files if used*

** This source code will be the one to be used in your demo (NO changes allowed after the submission).*

Your submission may include other files, such as `makefile` or a shell script for installation and a user guide.

In addition to your submission, **you need to run a demo** at an allocated workshop in Week 13.

Suggested headings for Stage 2 report (STRICTLY NO TITLE PAGE!)
(**max. 5 pages** including everything; the entire submission is 5 pages or fewer)

- **Project title**: no more than 100 characters (approximately 14 words), e.g., Cloud job scheduler or Cost-efficient resource allocator for distributed systems.
- Your name and SID, e.g., Young Lee (12345678).
- Introduction (½ page): What this project (focusing on Stage 2) is about, including the goal of Stage 2.
- Problem definition (½ page): the description of scheduling problem and the definition of your objective function including the justification of your choice.
- 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.
- Implementation details including data structure(s) used (½ page)
- Evaluation (2 pages): simulation setup including test cases/configurations, results, comparisons (with FF, BF and WF) and discussion including pros and cons of your algorithm.
- Conclusion (1/4 page): summary + what you have found and what you suggest
- References (1/4 page) including project git repository/wiki, e.g., GitHub and Bitbucket.

* The number of pages for each section is only indicative.

- Detailed demo instructions (e.g., user guide including installation manual) based on the "ready" VM we have been using for demos of stages 1 and 2 (no IDE specific).

**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. The performance of scheduling algorithm(s) needs to be extensively and thoroughly 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.

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. 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 (FF, BF and WF). 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.