Stage 3 (due 11:55pm Friday, 31st May 2020): the design and implementation of a new scheduling algorithm (or multiple algorithms)

This assessment item (Stage 3) accounts for 15%* of the total mark (100) of the unit. It is an individual work; hence, individual submissions. As usual, the marking will be conducted in three steps: (1) assessment of the submission (due Friday week 12), (2) demonstration of the work (at workshops in week 13) and (3) review and moderation of marks from (1) and (2).

By now, you should be fully aware how scheduling 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, as discussed in our lecture.

Your task in this stage is to design and implement at least one new scheduling algorithm that "optimises" one or more objectives. As there are multiple mostly conflicting/incompatible performance objectives in our scheduling, 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 execution costs (i.e., VM rental costs). Therefore, you have to define the scheduling problem clearly indicating your performance objective(s). You may present more than one scheduling algorithms of yours (of course) in the end, with description, comparisons and discussions between them although you may nominate one "champion" algorithm.

Your client-side simulator (or simply client) is required to recognise scheduling algorithms via the command line argument as in stage 2. i.e., the '-a' option.

Your client should work for any simulation configuration; do not assume those sample configurations given are the only ones used for testing and marking.

* The weighting has changed from 10% after combining previous weightings of 10% and 5% of stage 3 and final demo.

Deliverables

The submission is to be a **single compressed file** containing everything necessary to run simulations with your client containing your scheduling algorithm(s) and baseline algorithms and algorithm design document.

Three main components of your submission are as follows:
- The source code of your client-side simulator with scheduling algorithm(s)
- Algorithm design/implementation document
    - o Suggested headings (max 10 pages; no title page needed)
        - ▪ Project title: as in Stages 1 and 2
        - ▪ Your name and SID, e.g., James Zheng (88888888)
        - ▪ Introduction: What Stage 3 is about in your own words
        - ▪ Problem definition/formulation including the definition of your objective function
        - ▪ Algorithm description (one sub-section per algorithm if you have more than one); NB: You need to provid 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
        - ▪ Evaluation: simulation setup including test cases/configurations, results and discussion/justification
        - ▪ Conclusion: summary + what you have found and what you suggest
        - ▪ References (if any)
- 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).

Your submission may include other files, such as a shell script for installation and a user guide to allow others (including markers) to run simulations with your client.

Demo

In addition to your submission, you need to run a demo at an allocated workshop (as in stages 1 and 2) in week 13. You shall use detailed demo instructions you provided in your submission to do the demo.

Marking rubric (in percentage)

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 3 should be evident in both the submission and demo.

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 3 should be evident in both the submission and demo.

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 3 should be evident in the submission and demo, and the process of stage 3, e.g., git commit history.

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 3 should be evident in the submission and demo, and the process of stage 3, e.g., git commit history