Stage 2 (due 11:55pm Friday, 15th May 2020): the implementation of three baseline scheduling algorithms: First-Fit (FF), Best-Fit (BF) and Worst-Fit (WF)

This assessment item (Stage 2) accounts for **15%** of the total mark (100) of the unit. As it is to be a **group** work, each group shall make **only one submission** (no individual submissions). However, there are several components of Stage 2 to be done and marked individually (see the Deliverables section). The marking will be conducted in three steps: (1) assessment of the submission (due Friday week 10), (2) demonstration of the work (at workshops in week 11) and (3) review and moderation of marks from (1) and (2).

Your task in this stage is to implement three simple scheduling algorithms based on memory allocation policies in operating systems (i.e., FF, BF and WF). **Each member of a group** is required **to implement one 'distinct' algorithm** (for groups of 2, a total of two algorithms are required). These three scheduling algorithms will be used as baseline algorithms for Stage 3. The concise description of each of them in the form of a pseudo-code is shown below.

- **First-Fit**

  For a given job ji,
  1. Obtain server state information
  2. For each server type *i*, s_i , from the smallest to the largest
     3. For each server *j*, $s_{i,j}$ of server type s_i, from 0 to *limit* - 1 // *j* is server ID
        4. If server $s_{i,j}$ has sufficient available resources to run job j_i then
           5. Return $s_{i,j}$
        6. End If
     7. End For
  8. End For
  9. Return the first Active server with sufficient initial resource capacity to run job j_i

- **Best-Fit**

  For a given job ji,
  1. Set *bestFit* and *minAvail* to very large number (e.g., INT_MAX)
  2. Obtain server state information
  3. For each server type *i*, s_i , in the order appear in `system.xml`
     4. For each server *j*, $s_{i,j}$ of server type s_i, from 0 to *limit* - 1
        5. If server $s_{i,j}$ has sufficient available resources to run job j_i then
           6. Calculate the fitness value $fs_{i,j,}j_i$
           7. If $fs_{i,j,}j_i$ < *bestFit* or ($fs_{i,j,}j_i$ == *bestFit* and available time of $s_{i,j}$ <*minAvailTime*) then
              8. Set *bestFit* to $fs_{i,j,}$ j_i
              9. Set *minAvail* to available time of $s_{i,j}$
           10. End If
        11. End If
     12. End For
  13. End For
  14. If *bestFit* is found then
     15. Return the server with *bestFit*
  16. Else
     17. Return the best-fit Active server based on initial resource capacity
  18. End If

- Worst-Fit

  For a given job ji,
  1. Set *worstFit* and *altFit* to very small number (e.g., INT_MIN)
  2. Obtain server state information
  3. For each server type *i*, s_i , in the order appear in `system.xml`
     4. For each server *j*, $s_{i,j}$ of server type s_i, from 0 to *limit* - 1
        5. If server $s_{i,j}$ has sufficient resources readily available to run job j_i then
           6. Calculate the fitness value $fs_{i,j,}j_i$
           7. If $fs_{i,j,}j_i$ > *worstFit* and $s_{i,j}$ is immediately available then

8. Set *worstFit* to $fs_{i,j}, j_i$
9. Else If $fs_{i,j}, j_i$ > *altFit* and $s_{i,j}$ is available in a short definite amount of time then
10. Set *altFit* to $fs_{i,j}, j_i$
11. End If
12. End If
13. End For
14. End For
15. If *worstFit* is found then
16. Return the server with *worstFit*
17. Else If *altFit* is found then
18. Return the server with *altFit*
19. Else
20. Return the worst-fit Active server based on initial resource capacity
21. End If

** The fitness value of a job to a server is defined as the difference between the number of cores the job requires and that in the server.*

Note that if 'RESC All' is used, the available time of an Active server is always -1 regardless of its available resource capacity. In this case, if the server has sufficient available resources, -1 is interpreted as the current time, i.e., the server is available immediately.

Your client-side simulator (or simply client) is required to recognise scheduling algorithms via the command line argument as in the reference client, e.g., `./ds-client -a ff`. In particular, you are required to handle the '-a' option followed by the acronym of an algorithm as below. Note that the default algorithm is allToLargest (from stage 1) if no algorithm is specified.

- • `ff` for First-Fit
- • `bf` for Best-Fit
- • `wf` for Worst-Fit

Your client should work for any simulation configuration; do not assume those sample configurations given are the only ones used for testing and marking. Again, for testing and evaluation purposes, use the reference client-side simulator (`ds-client`) provided in `ds-sim.tar`.

For Stage 2, the simulation log generated by the server-side simulator (or simply server) may vary depending on how you implement your client. Therefore, comparisons between your client and the reference implementation (`ds-client`) may be conducted based on simulation statistics
presented at the end of simulation. In addition, simulation logs without using any verbose options (i.e., no `-v` options) might be helpful.

Deliverables

The submission is to be **a single compressed file** containing everything necessary to run simulations with implemented baseline algorithms (FF, BF and WF; again, two for groups of 2) and algorithm design document.
Three main components of your submission are as follows:

- The source code of your group's client-side simulator with baseline scheduling algorithms
  - o Each group member is required to create their own 'branch' (not an individual git repository) in the project git repository, for the implementation of their own algorithm.
  - o Your group may submit either a single client implementation containing all baseline algorithms (if merged successfully) or multiple implementations (one per group member if unable to merge).*
- Algorithm design document
  o Suggested headings (max 4 pages; no title page needed)
  - ▪ Project title: as in Stage 1
  - ▪ Group members: full names and student IDs, e.g., James Zheng (12345678)
  - ▪ Introduction: What Stage 2 is about in your own words
  - ▪ Design consideration and preliminaries: common data structures used and how resource/server state information is checked*
  - ▪ Algorithm description (one sub-section per algorithm; the group member who implemented a particular algorithm needs to be clearly indicated)**
  - ▪ References (if any)
- Detailed demo instructions (e.g., user guide including installation manual)*

* These sections may be marked individually if multiple implementations are submitted.
** This section is marked individually.

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

In addition to your submission, **your group needs to run a demo** at your allocated workshop in week 11. You shall use detailed demo instructions to do your demo.

Marking rubric (in percentage)
50 to 64
Work in this band presents the adequate design and implementation of baseline algorithms with appropriate algorithm description. The appropriate demonstration of teamwork and individual contributions should be evident.

65 to 74
Work in this band presents the good design and implementation of functional baseline algorithms with clear algorithm description. The appropriate demonstration of teamwork and individual contributions in all three assessment components should be evident.

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 fully functional baseline algorithms should be evident in (1) the source code, (2) design document and (3) demo. The sufficient demonstration of teamwork and individual contributions should be evident in all these three components, e.g., git commit history and participation in the demo.

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 baseline algorithms should be evident in (1) the source code, (2) design document and (3) demo. The clear demonstration of teamwork and significant individual contributions should be evident in all these three components, e.g., git commit history and all group members actively participating in the demo including confidently and sufficiently answering any questions from markers during the demo.