SIT323 Cloud Application Development, Trimester 2, 2020

Assessment Task 1 – Validation and Testing

Due Date

Sunday 8:00 PM, August 30, 2020

Introduction

Assessment Tasks 1 and 2 comprise parts of the one project. These require you to design, develop and test software related to a task allocation problem in which a parallel program is partitioned into a set of tasks and these tasks need to be allocated to a set of processors such the amount of energy consumed is minimised (see class notes of week 1).

1. In brief, Assessment Task 1 focuses on designing, developing and testing a new program such that it is able to:
    - Validate/invalidate the two data files related to configuration and task allocations.
    - Validate/invalidate allocations that are described in the allocations file.
    - Compute and display the amount of energy consumed by each allocation.
    - Display each allocation.

    You will be provided with valid input data files, but also invalid input data files. These files can be found in the ZIP file called **Assessment Task 1 – Data Files.zip**.

2. In brief, Assessment Task 2 requires you to focus on:
    - Designing, developing and testing a cloud solution.
    - Loading data from a configuration file to compute one or more allocations of tasks such that the amount of energy consumed is valid and is the lowest that you can achieve. Note, there might be more than one allocation where the consumed energy is the same and the lowest.
    - Optimisation techniques.
    - Computing and displaying the amount of energy consumed by these allocations.
    - Displaying these allocations.
    - Ensuring that these new allocations are valid.

Objectives

The main objectives for Assessment Task 1 are:

1. Design and develop a new program to work with the two input data files. Their formats are described in the following sections:
    - Task Allocation file format (.taff)
    - Configuration file format (.cff)
2. Design and develop many unit tests.
3. Code to conventions and standards.

Testing Requirements – Unit Tests

Your software solution for Assessment Task 1 requires testing. You must design and develop several unit tests for the following functionality.

1. Determining whether the amount of RAM required by a task is less than or equal to the amount of RAM associated with a processor.
2. Computing the runtime of a task allocated to a processor.
3. Computing the energy consumed by a task for running on a processor.
4. Computing the energy consumed by a task for local communications.
5. Computing the energy consumed by a task for remote communications.
6. Computing the total energy consumed by a task, i.e., the sum of the energies consumed related to the previous three points (3, 4 and 5).
7. Computing the total energy consumed by an allocation.

Task Allocations file format (.taff)

Your software solution for Assessment Task 1 must use data from a text file that contains zero or more task allocations (see "Test1.taff" as an example).

1. Lines containing zero or more white spaces only are allowed.

2. Lines containing a comment or data are permitted to commence with 0 or more white spaces.

3. A line containing a comment is allowed. The symbol // will indicate the start of a comment, the end of line will represent the end of a comment. Some valid comment lines are as follows.

```
// This is valid.
// Creation date: 31/12/2020
    // Leading white spaces are valid too.
```

   Mixing data and a comment on one line is not allowed. For example, the following line is invalid:

```
CONFIG-FILE="Test1.cff"      // Configuration filename.
```

4. There will be a line containing the file name of a configuration file. It commences with the keyword `CONFIG-FILE`, followed by an equals symbol, and ends with the filename that is delimited by double quotes. Two examples are below: the first is an absolute file name, the second is a relative file name.

```
CONFIG-FILE="C:\\temp\config.cff"
CONFIG-FILE=".\..\config.cff"
```

5. There will be a line containing data about allocations. It commences with the keyword `ALLOCATIONS-DATA`, followed by an equals symbol, the number of allocation in the file, the number of tasks in each allocation, and the number of processors in each allocation. For example, the following indicates 8 allocations, 5 tasks, and 3 processors. These 8 allocations are specified in a TAFF file and described below.

```
ALLOCATIONS-DATA=8,5,3
```

6. There will be a section of data for each allocation. In general, each allocation commences an allocation ID which is followed by a table representing the allocation of 0 or more tasks to each processor.

   For each allocation, there will be several lines of data. The first line commences with the keyword `ALLOCATION-ID`, followed by an equals symbol, and ends with an ID number.

   Following this line are several lines to specify the allocation, one line per processor. Each of these lines contain comma separated 1s and 0s such as 1,0,1,0,0 where values on the m^{th} line represents an allocation of 0 or more tasks to the m^{th} processor.
   - 1 in the n^{th} position of the m^{th} line indicates the n^{th} task is assigned to the m^{th} processor.
   - 0 in the n^{th} position of the m^{th} line indicates the n^{th} task is not assigned to the m^{th} processor.

   The following data represents one allocation with an ID of 3, 5 tasks, and 3 processors. Based on this data:
   - Processor 1 has been allocated tasks 1 and 4.
   - Processor 2 has been allocated tasks 2 and 3.
   - Processor 3 has been allocated task 5.

```
ALLOCATION-ID=3
1,0,0,1,0
0,1,1,0,0
0,0,0,0,1
```

Configuration file format (.cff)

Your software solution for Assessment Task 1 must use data from a CFF file that contains configuration data, an example can be seen in “Test1.cff”.

1. Lines containing zero or more white spaces only are allowed.

2. Lines containing a comment or data are permitted to commence with 0 or more white spaces.

3. A line containing a comment is allowed. The symbol // will indicate the start of a comment, the end of line will represent the end of a comment. Some valid comment lines are as follows.

```
// This is valid.
// Creation date: 31/12/2020
    // Leading white spaces are valid too.
```

   Mixing data and a comment on one line is not allowed. For example, the following two lines are invalid.

```
LIMITS-TASKS=1,100       // 1 to 100 tasks allowed.
LIMITS-PROCESSORS=1,500  // 1 to 500 processors allowed.
```

4. There will be a line containing the name of a log file. It commences with the keyword `DEFAULT-LOGFILE`, followed by an equals symbol, and ends with the filename that is delimited by double quotes.

   This can be an absolute or a relative file name. For example:

```
DEFAULT-LOGFILE="log-2020.txt"
```

5. There will be a section of minimum and maximum limits for the number of tasks, the number of processors, the processor frequencies, and amounts of RAM.

   Each line in this section commences with a keyword, following by an equals symbol, a minimum value, another comma, and ends with a maximum value. For example:

   ```
   LIMITS-TASKS=1,500
   LIMITS-PROCESSORS=1,1000
   LIMITS-PROCESSOR-FREQUENCIES=1.2,10.0
   LIMITS-RAM=1,64
   ```

   This means that we cannot have a program partitioned into more than 500 tasks, we cannot use more than 1000 processors, we cannot use a processor that has a frequency of more than 10 GHz, and we cannot allocate more than 64 GB of RAM to a processor, etc.

6. There will be a line containing data related to the parallel program. It will commence with the keyword `PROGRAM-DATA`, following by an equals symbol, the maximum duration of that program, the number of tasks that this program must be partitioned, and the number of available processors to run the tasks. For example:

   ```
   PROGRAM-DATA=3.0,5,3
   ```

   This means that the parallel program must complete within 3 seconds. It must be partitioned into 5 tasks. Each task must be allocated to one of the 3 processors.

7. There will be a line indicating the frequency of a reference processor. The runtime of a task is based on executing that task on a processor running at that frequency. This line commences with the keyword `REFERENCE-FREQUENCY`, following by an equals symbol, and ends with a frequency value (in GHz). For example:

   ```
   REFERENCE-FREQUENCY=2.0
   TASK-RUNTIME-RAM
   ```

8. There will be a line containing a runtime and RAM values. This line commences with the keyword `TASK-RUNTIME-RAM`, following by an equals symbol, followed by a sequence of pairs (runtime and RAM).

   For example:

   ```
   TASK-RUNTIME-RAM=1.0,2,1.0,2,2.0,4,1.0,2,3.0,8
   ```

   This means that:

   - task 1 has a runtime of 1 second and requires 2 GB of RAM
   - task 2 has a runtime of 1 second and requires 2 GB of RAM
   - task 3 has a runtime of 2 second and requires 4 GB of RAM
   - task 4 has a runtime of 1 second and requires 2 GB of RAM
   - task 5 has a runtime of 3 second and requires 8 GB of RAM

9. There will be a line containing processor type, frequency and RAM values. This line commences with the keyword `PROCESSORS-FREQUENCIES-RAM`, following by an equals symbol, followed by a sequence of triples (type, frequency and RAM).

   For example:

   ```
   PROCESSORS-FREQUENCIES-RAM=Intel i5,1.8,4,AMD
   EPYC,2.4,4,Intel i9,3.6,8
   ```

This means that:

- processor 1 is an Intel i5 running at 1.8 GHz with 4 GB of RAM
- processor 2 is an AMD EPYC running at 2.4 GHz with 4 GB of RAM
- processor 3 is an Intel i9 running at 3.6 GHz with 8 GB of RAM

10. There will be a line containing processor types and coefficients of a quadratic formula. This line commences with the keyword `PROCESSORS-COEFFICIENTS`, following by an equals symbol, followed by a sequence of type and coefficient values.

    For example:

    ```
    PROCESSORS-COEFFICIENTS=Intel i5,25,-25,10,Intel i9,25.0,-26.0,9.5,AMD EPYC,24.0,-25.0,10.0
    ```

    This means that:

    - the Intel i5 processor has an energy consumption function of $10f^2 - 25f + 25$
    - the Intel i9 processor has an energy consumption function of $9.5f^2 - 26f + 25$
    - the AMD EPYC processor has an energy consumption function of $10f^2 - 25f + 24$

    This means that a task that runs for 2.5 seconds on a 3.3 GHz Intel i5 processor will consume the following amount of energy.

    $$(10*3.3^2 – 25*3.3 + 25) * 2.5$$
    $$= (10*3.3^2 – 25*3.3 + 25) * 2.5$$
    $$= (108.9 – 82.5 + 25) * 2.5$$
    $$= 51.4 * 2.5$$
    $$= 128.5$$

11. There will be a section containing energy costs for local communications. This section commences with the keyword `LOCAL-COMMUNICATION` on a line by itself, this is followed by a table of energy values.
    - The number of rows is the same as the number of tasks.
    - The number of columns is also the same as the number of tasks.
    - Table element(N, M) represents the amount of energy required to send data from Task N to Task M on the same processor.
    - When element(N, M) is zero, Task N never sends data to Task M.
    - When element(N, M) is non-zero, Task N does send data to Task M.

    For example, the following data indicates that:

    - Task 1 sends data to Task 2, 3, 4 and 5.
    - Task 2 sends data to Task 3.
    - Task 3 sends data to Task 4.
    - Task 4 sends data to Task 5.
    - Task 5 sends data to Task 1.

    ```
    LOCAL-COMMUNICATION
    0,0.0001,0.0001,0.0001,0.0001
    0,0,0.00005,0,0
    0,0,0,0.00005,0
    0,0,0,0,0.00005
    0.0001,0,0,0,0
    ```

12. There will be a section containing energy costs for remote communications. This section commences with the keyword `REMOTE-COMMUNICATION` on a line by itself, this is followed by a table of energy values.
    - The number of rows is the same as the number of tasks.
    - The number of columns is also the same as the number of tasks.
    - Table element(N, M) represents the amount of energy required to send data from Task N running on a processor to Task M running on another processor.
    - When element(N, M) is zero, Task N never sends data to Task M.
    - When element(N, M) is non-zero, Task N does send data to Task M.

    For example, the following data indicates that:
    - Task 1 sends data to Task 2, 3, 4 and 5.
    - Task 2 sends data to Task 3.
    - Task 3 sends data to Task 4.
    - Task 4 sends data to Task 5.
    - Task 5 sends data to Task 1.

```
REMOTE-COMMUNICATION
0,0.1,0.1,0.1,0.1
0,0,0.05,0,0
0,0,0,0.05,0
0,0,0,0,0.05
0.1,0,0,0,0
```

Validating files and allocation

1. Your software solution for Assessment Task 1 must be able to validate both text input data files (taff and cff).

   Invalid aspects of these files are displayed to a user via the GUI.

2. Your software solution for Assessment Task 1 must be able to validate allocations too, but only if both input data files are valid. Consequently, it is possible to have perfectly valid input files, but an allocation is invalid.

   For example, an allocation is invalid if the accumulated runtime of tasks exceeds the overall program runtime.

   As another example, the following allocation is invalid because task 1 has been allocated to two processors.

```
1,1,0,0,0
0,0,1,1,0
1,0,0,0,1
```

   Invalid aspects of an allocation should also be displayed on the GUI.

Displaying an allocation and the amount of energy consumed

1. Whether or not the input files and allocations are valid, the existing program attempts to display valid or invalid allocations on the GUI. For example, the invalid allocation in the above section could still be displayed.

2. The existing program computes and displays the amount of energy consumed by a valid allocation on the GUI. However, an appropriate message is displayed for an invalid allocation.

Data files

To help you design and develop your unit tests and software, you will be provided with several pairs of CFF and TAFF files based on the above formats. These files can be found in the ZIP file called **Assessment Task 1 – Data Files.zip**.
