Faculté des Sciences
Département d'Informatique

INFO-F-404 – Real-Time Operating Systems

Scheduling project

Partitioned EDF

22 octobre 2020

1 Project objectives

In this project, you will be required to write a fixed job priority scheduler in a partitioned identical parallel systems that checks the feasibility of a set of n tasks on m identical processors and then simulates that execution.

2 Project details

This project is individual.

2.1 Programming language

You get to choose the programming language of the project between Python3 and Rust but be careful to structure and document your code properly so that it is understandable.

2.2 Program inputs

Your program will accept multiple parameters and options on the command line as shown in the below example.

```
$ ./partitioned_edf <tasks_file> -h ff|wf|bf|nf -s iu|du -l <limit> -m <#cores>
```

2.2.1 Task set file

The task set file is a file that contains a description of the tasks that have to be scheduled. This file has four columns separated by white-spaces that represent the offset, WCET, deadline and period. An example is shown below.

```
0 10 50 50
0 20 80 80
0 10 100 100
0 50 200 200
```

You can implement your own task generator to perform your tests. It would for instance take the number of tasks and the utilisation of the taskset as arguments.

2.2.2 Program options

-h The `-h` option (for heuristic) is mandatory and can take one of the values in `ff|wf|bf|nf` which respectively stand for *first fit*, *worst fit*, *best fit* and *next fit*.

-s The `-s` option (for sort) is mandatory and can take one of the values in `du|iu` which stand for *increasing utilisation* and *decreasing utilisation*.

-l The `-l` option (for limit) is mandatory and can take any positive integer value. It states the time step limit for the simulator, i.e. the time step until which the simulator has to run.

-m The `-m` option can take any positive integer value and indicates the number of identical cores.

3 Partitioner

In this project, you will be required to implement a partitioner whose purpose is to partition the task set between the different processors according to a combination of sorting order and heuristic.
For instance, you might be required to use the FF (first fit) and DU (decreasing utilisation) combination to partition the tasks among the cores.

4 EDF Scheduler

You have to implement an scheduler that can schedules the jobs of a core with the EDF (earliest deadline first) strategy.
Feel free to implement other schedulers as well and to document them in your report.

5 Program execution example

The program should use the partitioner to partition a taskset among the cores and then simulate the execution with the EDF scheduler. This simulation should give an overview of what is happening and can either be graphical or textual.

Here is an **example** of a program execution. Please remind that this is just an example and that you can adapt it to your needs. If you want to use a GUI or a web server, feel free to do so as long as it is documented in your report.

In this example, there is one processor and we use the same taskset as the example in Section 2.2.1. This example has been launched with the following command

```
$ ./partitioned_edf taskset.txt -h ff -s du -l 150 -m 1
```

And gives the following output.

```
0       P0      T0J1 released
0       P0      T1J1 released
0       P0      T2J1 released
0       P0      T3J1 released
0-9     P0      T0J1
10-29   P0      T1J1
30-39   P0      T2J1
50      P0      T0J2 released
50      P0      Deadline of T0J1
80      P0      T1J2 released
80      P0      Deadline of T1J1
```

```
40-89    P0      T3J1
90-99    P0      T0J2
100      P0      T0J3 released
100      P0      T2J2 released
100      P0      Deadline of T0J2
100      P0      Deadline of T2J1
100-109  P0      T0J3
110-129  P0      T1J2
130-139  P0      T2J2
```

6 Report

You have to write a short report that contains at least

- a short description of what the report is about (the project, the context, the scope, ...)
- all you have done in addition to the requirements of this document (additional options, task generator, additional scheduler, ...)
- a description of your implementation choices (diagrams, structure justification, ...)
- a section where you describe difficulties that you met during this project and how you overcame them
- anything you think is relevant to show the work you have provided

7 Evaluation criteria

This project will be rated out of 20 points and aims at evaluating your understanding of the course theory in the field of parallel architectures. You will be evaluated on

- The partitioning algorithms are correctly implemented /5
- The EDF scheduling algorithm works as expected /5
- The report /6
  - contains relevant information /4
  - is well structured /1
  - contains a user manual /1
- The code is readable, well structured and properly documented /4

Total /20

Any implementation in addition to the strict requirements of this sheet can grant you additional marks.

8 Handing in your project

You have to hand in a zip file containing at least

- your source code
- your report in PDF

You have to upload this zip file on the "Université Virtuelle" no later than the 19th of November, 23:59.

Delays

Late projects will also be submitted on the UV with a penalty of 1 point for every 4h with at most 24h of delay.