Parallel computing - Exercise 2

Michela Venturini

Spring 2019

1 Parallelize code by using OpenMP

The task of the exercise consists in paralelizing a given chunk of code to visualize OpenMP scheduling of the threads. The scheduling of the *for loop* of the code is specified by the option `#pragma omp for schedule([type_of_scheduling]) private(i)`. The scheduling can be static or dynamic and it is also possible to specify the size of the chunk being assigned to each thread.

static The static option impiles that iterations blocks are mapped statically to the execution threads in a round-robin fashion.

dynamic The dynamic sheduling works on first-come first-served policy. This implies that different executions of the same code with the same number of threads may produce different results. This strategy can lead a better workload balancing w.r.t. the previuous one but might introduce some additional overhead.

2 Execution

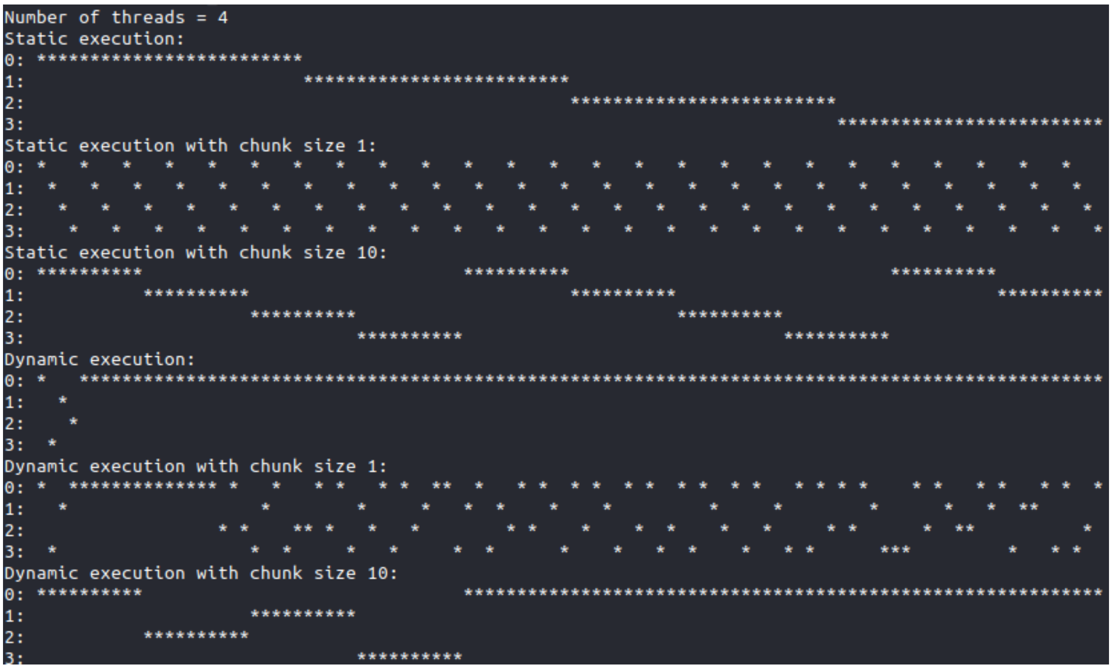
The code is executed on Ulysses by using the options:

1. static
2. static, with chunk size 1
3. static, with chunk size 10
4. dynamic
5. dynamic, with chunk size 1
6. dynamic, with chunk size 10

The execution is performed by submitting a job on Ulysses through the command `qsub nodes=1:ppn=20 ex2.sh` that asks for a single node, sufficient for this purpose.

3 Results

The result of executions are presented in the 1. As expected, the behaiviour of `dynamic` executions leads to a different result in the scheduling of the threads that is not fixed as in the other case.



```
Number of threads = 4
Static execution:
0: *************************
1:                          *************************
2:                                                   *************************
3:                                                                            *************************
Static execution with chunk size 1:
0: *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
1:  *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
2:   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
3:    *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
Static execution with chunk size 10:
0: **********                              **********                              **********
1:           **********                              **********                              **********
2:                     **********                              **********
3:                               **********                              **********
Dynamic execution:
0: *   ************************************************************************************************
1:   *
2:    *
3:  *
Dynamic execution with chunk size 1:
0: *  ************** *   *   * *   * *  **  *   * *  * *  * *  * *  * *   * * * *    * *   * *   * *  *
1:   *                  *        *     *   *  *    *    *         *     *        *      *   *  **
2:                  * *    ** *   *   *        * *    *    *  *    *   *     * *      *  **          *
3:  *                  *  *     *   *     *  *      *    *   *  *    *   * *      ***         *   * *
Dynamic execution with chunk size 10:
0: **********                              ************************************************************
1:                     **********
2:           **********
3:                               **********
```

Figure 1: Result of the executions with 4 threads