# PARALLEL AND DISTRIBUTED COMPUTING

2016/2017 — 2nd Semester

2nd Exam — July 3rd, 2017 — Duration: 2h00

- No extra material allowed. This includes notes, scratch paper, calculator, etc.
- Give your answers in the available space after each question. You can use either Portuguese or English.
- Be sure to write your name and number on all pages, **non-identified pages will not be graded!**
- Justify all your answers.
- Do not hurry, you should have plenty of time to finish this exam. Skip questions that you find less comfortable with and come back to them later on.

**I. (1,5 + 1 + 1 + 1,5 = 5 val.)**

1. Write down two possible outputs for the following program. Assume that each thread only flushes the `stdout` when the newline character is printed.

```
#include <stdio.h>
#include <omp.h>

int main(int argc, char* argv[])
{
    int i;
    #pragma omp parallel num_threads(2)
    {
        printf("X");
        #pragma omp for schedule(static,2)
        for(i = 0; i < 7; i++)
        printf("%d", i);
        #pragma omp single
        printf("Y");
        printf("\n");
    }
}
```

2. What is the effect of the `#pragma omp parallel` directive and what is its scope?

3. Discuss the differences between `#pragma omp critical` and `#pragma omp atomic`.

4. The Sieve of Erastothenes is a simple algorithm for generating prime numbers that creates a sequence of all the integers up to a given number $n$, picks one number $p$ from the sequence, in ascending order, removes all the multiples of $p$ above $p^2$ and repeats until it reaches the end of the sequence. Write an efficient OpenMP-based parallel implementation of the function `int se(int v[], int n)`, listed below, just by introducing OpenMP directives or block delimiters (`{}`) wherever necessary. Upon the execution of the function, vector `v` should contain all the prime numbers computed, in any order.

```
int se(int v[], int n)

{

    int *m = calloc(n, sizeof(int));

    int i, p = 2, t = 0;

    while(p*p < n) {

        for(i = p*p; i < n; i += p)

            m[i] = 1;

        for(p++; m[p]; p++);

    }

    for(i = 2; i < n; i++)

        if(!m[i])

            v[t++] = i;

    free(m);

    return t;

}
```

**II. (1,5 + 1,5 + 2 = 5 val.)**

1. In an optimized implementation of the MPI function `MPI_Gather`, what is the size of the last message the destination process receives? Explain. (assume $p$ represents the number of processes and $n$ the complete size of the array)

2. Steps 3 and 4 of the Foster's design methodology are, respectively, "Aggregation" and "Mapping". In what way are they similar, and what is the fundamental difference?

3. Consider the following piece of code, where:

   - variable `id` holds the identifier of the MPI task;
   - array `words` contains the words to be searched in array `text` (assume each tasks uses a different text);
   - array `hits` gets the number of times each word is found in the text;

```
if(id != 0) {
    hits = find_words(words, N_WORDS, text[id]);
    MPI_Send(hits, N_WORDS, MPI_INT, 0, COUNT, MPI_Comm_world);
}
else {
    total = find_words(words, N_WORDS, text[id]);
    for(i = 1; i < NUM_PROCS; i++) {
        MPI_Recv(hits, N_WORDS, MPI_INT, i, COUNT, MPI_Comm_world, &status);
        for(j = 0; j < N_WORDS; j++)
            total[j] += hits[j];
    }
}
```

   Propose an optimized MPI implementation for this code. (don't worry about the syntax of any MPI routine you use, but make sure all the relevant parameters are there)

**III. (1,25 + 1,25 + 2,5 = 5 val.)**

1. What is the meaning of obtaining 0 for the *Experimentally determined serial fraction* metric, when evaluating a program on 2, 4, 8 and 16 processors.

2. Discuss the limitations of Amdahl's Law.

3. Suppose that for an application:

   - the serial cost execution time is $\Theta(n^3)$
   - the parallel computation time is $\Theta(n^3/p)$
   - the memory requirements are $\Theta(n^2)$
   - the communication overhead is $\Theta(n^2 \log p)$

   where $n$ is the size of the problem and $p$ is the number of processors. Determine how scalable this application is, based on the scalability function. Justify your answer.

**IV. (1 + 1 + 1 + 1 + 1 = 5 val.)**

1. The typical approach for a parallel implementation of a Parallel Backtrack Search algorithm is to go down a few levels of the search tree and then distribute the subtrees at that level to the different tasks. State the advantages and disadvantages of going to a deeper level of the tree before distributing the work.

2. Many problems involving the simulation of physical systems are based on finite difference methods.

   a) Briefly describe what are finite difference methods.

   b) These methods generally require the use of *ghostpoints*. Explain why.

3. In Monte Carlo methods, discuss the problems with a centralized approach for the random number generation in:

    a) a shared-memory implementation.

    b) a distributed-memory implementation.