PARALLEL AND DISTRIBUTED COMPUTING

2017/2018 2nd Semester

1st Exam June 15th, 2018 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 + 1.5 + 1 + 1 = 5 val.)

1. In the `#pragma omp parallel for` directive, the `schedule` clause enables the specification of how the iterations of a loop should be scheduled, that is, allocated to threads. Describe how each of the following scheduling types work and their parameters (for the ones that have): `static`, `dynamic`, `guided`. For those scheduling types that have optional parameters, you should describe the behavior in the presence and absence of such parameters.

2. Consider the function `int ones(int* m, int n)` which receives a pointer `m` to the first element of a square matrix and its number of rows/columns `n`, and computes que number of elements with value `1`. The matrix is organized in memory in row major order (i.e. by rows). Each element of the matrix can only assume the value `0` or `1`. Provide the best parallel implementation for this function using OpenMP.

3. What are the negative effects of false sharing, associated to shared-memory systems, and how can this problem be dealt with?

4. What is the behavior of `#pragma omp parallel if (<expression>)`? When should it be used?

II. (1 + 1.5 + 1 + 1.5 = 5 val.)

1. What is the difference, in terms of behavior and usage, between the `MPI_Recv` and `MPI_Irecv` methods? Under what circumstances should one use either of them?

2. Let A be a large array of N single digit integer values, i.e. varying between 0 and 9 inclusive. Write an MPI routine that efficiently computes the mode, i.e. the most frequently occurring value, among the values in A and prints the result. Assume that, initially, A is only available to the process with rank 0. For simplicity, you may also assume that A can be evenly distributed among the processes, i.e. $\frac{N}{P} = C$ where P is the number of processes and C is some integer constant. Explicitly include in your code the data transfer routines (don't worry about the exact MPI syntax, but be sure to specify all the relevant parameters).

3. Consider the following MPI code:

```
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int** a = (int**)malloc(N * sizeof(int*));
for (int i = 0; i < N; ++i) {
    a[i] = (int*)malloc(N * sizeof(int));
}
if (rank == 0) {
    read(a, N);
}
for (int i = 0; i < N; ++i) {
    MPI_Bcast(a[i], N, MPI_INT, 0, MPI_COMM_WORLD);
}
```

a) The code above is very inefficient. Explain why.

b) Rewrite the code above to make it as efficient as you can.

III. (1.5 + 1 + 1 + 1.5 = 5 val.)

1. The sequential implementation of an algorithm runs in $\Theta(n^2 \log n)$. The computation time of its parallel version is $\Theta(\frac{n^2 \log n}{p})$. The parallel overhead (communication+ synchronization+redundant computations) is $\Theta(n \log n \log p)$. The required memory increases with n^2.

   a) Compute the isoefficiency relation for this parallel implementation. Explain the meaning of the isoefficiency relation.

   b) Compute the scalability function and discuss whether this implementation is scalable of not.

2. Consider that for the sequential execution of a given program the relation between the time spent executing purely sequential code and the time spent executing completely parallelizable code is α. Assuming that all the code is either purely sequential or completely parallelizable, and according to Amdahl's Law, what is the maximum speedup achievable by parallelizing this program?

3. Benchmarking a parallel program on 1, 2, . . ., 8 processors produces the following speedup results:

p	2	3	4	5	6	7	8
ψ	1.87	2.61	3.23	3.73	4.14	4.46	4.71

What is the experimentally determined serial fraction, proposed by Karp and Flatt? Resorting to this metric, and explaining your reasoning, discuss what is the main factor that is hurting the parallelization.

IV. (1 + 1.5 + 1 + 1.5 = 5 val.)

1. In the Backtrack Search algorithm a depth-first search on a tree is usually used. However, in a parallel implementation, in general, all tasks perform the expansion of the first levels of the tree, which leads to redundant operations. What is the reason for this apparent wasteful computation?

2. What is the limitation of Hyperquicksort that the Parallel Sorting by Regular Sampling algorithm tries to address? How do these algorithms compare in terms of scalability?

3. Is it possible to obtain an efficiency $\varepsilon > 1$ in a parallel system? Explain.

4. Consider the Heat Distribution problem studied in class. What is the purpose of using ghostpoints? Explain the compromise that is being targeted.