PARALLEL AND DISTRIBUTED COMPUTING

2018/2019 — 2nd Semester

1st Exam — June 14th, 2019 — 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 + 1 + 1 + 1 + 1 = 5 val.)

1. Give one possible output of the following program:

```
void func(int id)
{
#pragma omp for
    for(int i = 0; i < 4; i++)
        printf("Tid: %d\ti = %d\n", id, i);
}

int main()
{
#pragma omp parallel num_threads(2)
    {
        int tid = omp_get_thread_num();
        printf("Thread %d alive!\n", tid);
#pragma omp single
        {
            printf("Thread %d in single!\n", tid);
            func(tid);
        }
    }
    printf("All done!\n");
}
```

2. Explain the difference between *memory consistency* and *memory coherence*.

3. Consider the following code:

```
int i,j;
#pragma omp parallel for
for(i = 0; i < 10; i++){
  for(j = 0; j < 10; j++){
    array[i] += buffer[i*10 + j];
    array[j] -= buffer[i*10 + j];
  }
}
```

   a) Explain the meaning of a race condition.

   b) Explain why the code above has a race condition.

   c) The following change was introduced to remove the race condition:

```
int i,j;
#pragma omp parallel for
for(i = 0; i < 10; i++){
   for(j = 0; j < 10; j++){
      array[i] += buffer[i*10 + j];
      #pragma omp critical
      {
        array[j] -= buffer[i*10 + j];
      }
   }
}
```

   Why did this change solved (or not solved) the problem? Justify.

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

1. What is the difference between `MPI_Test` and `MPI_Wait`?

2. Explain the functionality of `MPI_Bcast`. For 8 processes, draw an illustration and explain how information is transmitted between each of the processes, during the execution of this function, assuming an optimized implementation.

3. Consider the function `MPI_Alltoall` presented below.

```
int MPI_Alltoall(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
                 void *recvbuf, int recvcount, MPI_Datatype recvtype,
                 MPI_Comm comm)
```

Provide an implementation for `MPI_Alltoall`, using other MPI functions (with the exception of `MPI_Alltoallv`). Even a simple, but correct, implementation is acceptable.

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

1. Comment the following statement, saying whether you agree with it or not, and why.

   “For a given parallel execution of a program, we can obtain a speedup value using the Amdahl’s Law and a different speedup value using the Gustafson-Barsis’ Law".

2. When using 4 processors, the measured speedup over the sequential execution time of a program is 2. Assuming that the inefficiency of the parallel program is dominated by a large serial fraction, indicate a reasonable estimate for the speedup on 8 CPUs. (hint: use the Karp-Flatt Metric)

3. The isoefficiency relation for a given parallel implementation of an application has been determined to be $n > C\sqrt{p}$, where n is the problem size, p the number of processes and C a scalar constant.

   a) State in your own words what is the meaning of this expression.

   b) If the memory required by the program grows with n^3, what is the scalability function of this implementation? Is this a scalable solution?

IV. (1,25 + 1,25 + 1,25 + 1,25 = 5 val.)

1. When using dynamic load management, should you execute first larger or smaller tasks? Justify.

2. Consider a work-pool programming model. Give a short piece of pseudocode that implements the operation of the master.

3. In a Parallel Branch and Bound Search implemented on a distributed system, discuss the relative advantages of having a single priority queue versus multiple priority queues (one at each node). What's the best solution?

4. In the Ring Termination algorithm, explain why it may be necessary for the token to go through each process twice.