PARALLEL AND DISTRIBUTED COMPUTING

2017/2018 — 2nd Semester

2nd Exam — July 2nd, 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 + 1 + 1.5 = 5 val.)

1. Consider the function `weightedSum`, with the following prototype:

```
double weightedSum(double* v, int nv, double* w, int nw)
```

This function computes a weighted sum of the elements of vector `v`, where the weights are given by vector `w`. Vector sizes are `nv` and `nw`, respectively. If `nw` < `nv`, then `w` should be repeatedly applied to `v`. For example, if `nv=4` and `nw=2`, then the value returned by the function should be `v[0]*w[0]+v[1]*w[1]+v[2]*w[0]+v[3]*w[1]`. Write down a parallel implementation for this function based on OpenMP. Vectors `v` and `w` should not be modified. Only the most efficient implementations will get full credit.

2. Write down one possible output for the following program.

```
int main(int argc, char* argv[])
{
    int i, j, m;
    #pragma omp parallel private(m,i) num_threads(4)
    {
        for(i = 1; i < 5; i++) {
            m = i;
            #pragma omp for
            for(j = 0; j < i; j++)
                m += j;
            printf("%d ", m);
        }
    }
    printf("\n");
    return 0;
}
```

(numbers by any order on each word)

3. What is the effect of using the `nowait` clause with the `for` or `sections` directives of OpenMP?

4. In shared-memory systems, caches usually implement a snooping protocol. Describe what it consists of and why it is necessary.

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

1. What is their functionality and the difference, in terms of behavior and usage, between the `MPI_Wait`, `MPI_Waitall` and `MPI_Waitany` methods? Under what circumstances should one use either of them?

2. Let A be a large array of N integer values. Write an MPI routine that implements the functionality of `MPI_Alltoall` using `MPI_Scather` (do not worry about the exact MPI syntax, but be sure to specify all the relevant parameters). For simplicity, you may 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. Only the most efficient solutions will get full credit.

3. Consider the following MPI code:

```
int rank, p;
int a[N], b[N];
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &p);
if (rank == 0)
    read(a, N); /* process 0 reads array from file */
MPI_Bcast(a, N, MPI_INT, 0, MPI_COMM_WORLD);
for (i = N/p * rank; i < N/p * rank+1; ++i) {
    a[i] = someComputation(a[i]);
    MPI_Barrier(MPI_COMM_WORLD);
}
MPI_Gather(a + N/p * rank, N/p, MPI_INT, b, 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. Only the most efficient solutions will get full credit.

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

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

   a) Compute the isoefficiency relation for this implementation.

   b) Compute the scalability function and, explaining what it means, discuss whether this implementation is scalable or not.

2. Consider that an application running on 10 processors spends 10% of the time executing parallel code. Compute the scaled speedup.

3. Explain what is the experimentally determined serial fraction, proposed by Karp and Flatt. Detail, step by step, the procedure that a programmer should follow to optimize a parallel implementation guided by such metric.

   Procedure

   1. Compute $\Phi(n, p)$ for several values of p and n large enough.
   2. Compute $e = \frac{\frac{1}{\Phi(n,p)} - \frac{1}{p}}{1 - \frac{1}{p}}$ from $\Phi(n, p)$
   3. If e remains constant with p, large serial fraction is the reason for limited speedup. If e steadily increases with p, overhead is the reason for limited speedup.
   4. Fix the reason for limited speedup.

IV. (1 + 1 + 0.5 + 1.5 + 1 = 5 val.)

1. Two programming models used with distributed memory systems are Distributed Shared Memory (DSM) and Message-Passing. Describe the main differences between them and discuss the relative merits.

2. Foster's design methodology includes two steps designated by "Aggregation" and "Mapping". In what way are they similar, and what is their fundamental difference?

3. In one of the lectures we have studied a termination detection algorithm designated by Ring Termination Algorithm. Consider the version of the algorithm that is able to deal with reactivation.

   a) On what parallel implementations should this algorithm be employed?

   b) Explain how the algorithm works. Do not forget to mention: how communication is organized, what is the role of the token, how it is propagated and what are the conditions that ensure proper termination

4. Software Transactional Memory (STM) has been proposed as an alternative to lock-based synchronization. Enumerate two benefits and two drawbacks of STM, explaining why you consider each of them to be either a benefit or a drawback.