Introduction to High-Performance Computing

1st session — 20 May 2022 — Duration : 2 hours.

All documents are allowed.

Like in all exams, your answers must come with the necessary explanations.

This exam revolves around a (seemingly) simple problem : computing, for many different values of the first term, the Nth term of a sequence that is defined by a recurrence equation of the form

$$x_{i+1} = F\left(x_i, a_i\right).$$

The first term x_0 is known ; the *auxiliary values* $a_0, a_1, \ldots$ that are necessary to compute the sequence terms are also known. The goal is to determine whether the Nth term of the sequence has some particular property. This is quite unlikely to happen, so many attempts must be made. In particular, we assume that the number of attempts is significantly larger than N. A sequential program solving this problem is given in the supplemental document.

Exercice 1 – Distributed version with MPI

1. May the function `check()` be parallelized ? (If yes, how ? If no, why ?)
2. Using MPI, write C code for a modified version of the function `find()` which computes p sequences in parallel, where p is the number of available MPI processes.
3. Is there any risk of load balancing issues ?

Hints :

— In a first phase, we try sequences $0, 1, \ldots, p-1$, then a second phase we try the sequences $p, p+1, \ldots, 2p-1$, etc.

— The difficulty is to stop the computation in a synchronized manner when a solution is found. A collective operation will be useful for this.

Exercice 2 – Multithread parallelization with OpenMP

1. Using OpenMP, write C code for a modified version of the function `find()` which computes t sequences in parallel, where t is the number of available threads.
2. If one only has one computing server available, does this version have any interest compared to the MPI version ?
3. Can we combine both ? (If yes, how ? If no, why ?)

Some reminders :

— One cannot use `return` from inside a parallel region.

— Inside a parallel region, the functions `omp_get_num_threads()` and `omp_get_thread_num()` can be used to obtain respectively the total number of threads (t) and the number of the current thread (between 0 and $t-1$).

Exercice 3 – "*big-data*" version with MPI

Now assume that N is very large and the array of auxiliary values cannot anymore be stored in the memory of a single computing server (for example, it can require a few TeraBytes...). This array (`aux` in the provided code) will be itself distributed between the processes.

1. Choose a method to distribute the `aux` array between the processes (explain your choice), and write C code for the two following functions using an MPI library.

```
/*
 * Returns a part of the auxiliary array.
 * All processes call this function at the beginning of main().
 */
T * distributed_setup();

/*
 * Does the same thing as check().
 * All processes must call this function.
 * The parameter aux must be the return value of distributed_setup().
 */
bool distributed_check(long j, long N, T * aux);
```

2. Assume that the transmission of a message of length n takes a time $\alpha + \beta n$. Assume also that every call to `update` take a time γ. What is the global execution time ?

Exercice 4 – "*big-data*" version with a pipeline

With the array `aux` distributed as in the previous section, we would like to have code which tests many sequences in parallel, as fast as possible. For this, we can use a strategy with a *pipeline*. The general idea is that the process of rank i receives a set of k terms (coming from k sequences) from its predecessor ; it makes progress in these k sequences using `update` with its part of the array `aux` ; and it transmits the k obtained values to its successor. The process of rank 0 initializes the sequences using `initialize()` ; the last process calls `is_special()`. Of course, they all do this *simultaneously*.

1. How to handle the termination ?
2. Write a parallel version, using MPI, in C or pseudocode, which solves the problem as fast as possible.
3. Could one try to improve performance by overlapping communication and computation ?
4. In this precise case, what is the network characteristic which has the greatest impact ? Its latency, or its throughput ? (explain your answer)
5. Estimate execution time depending on the number of cores C of each computing server, the number p of computing servers, and the size ℓ of an element of the sequence. How to choose k in an optimal way ?
6. If $\gamma = 0.1\mu s$, and one has 64 computing servers each with 32 cores, and one element of the sequence uses 16 bytes, what should be the network characteristics so that the computation runs at optimal speed ?