Assignment #2 - MPI & Computation Graphs

TASK #1

Why do we need to perform the ring twice?

We have to perform the ring twice because, in the first turn, all the processes declare “barrier reached” to the near one, and in the second, they acknowledge “barrier passed” and continue with their normal flow.

What is the complexity of the function "My_Barrier"?

The time complexity is O(N)

TASK #2

What completion time do you expect for the computation of the stream?

The function "IN" requires 4ms to calculate each element of the array. The array length is 10, "IN" passes a new input to "M" every 40ms. The function "OUT" takes 1ms for each element of the array; the array's computation needs 10ms. The function "M" applies the function g1(f1(x[i]))=((x[i]*2)/2) where x[i] is a single element of the array. "f1" takes 12ms and "g1" 8ms, "M" takes 20ms to calculate a single element, an array is calculated in 200ms. Because of this, M is a bottleneck, obligates "IN" to wait before sending the next input, and forces "OUT" to wait to receive the successive output. If M wasn't a bottleneck, the computation of the stream should take approximately the time needed by "IN" to pass all inputs (40ms*100input=4000ms=4seconds). But in our case, the complete computation needs approximately the time used by "M" to compute the stream (200ms*100input=20000ms=20seconds).

What is the value ρ (i.e., the traffic intensity) of the module M?

Out system is composed of 3 modules "M" , "IN" and "OUT" with latency 200ms, 40ms, and 10ms respectively. We apply the formula: Ta1=40/1 = 40ms ρ = 200/40= 5 > 1

TASK #3

What completion time do you expect for the computation of the stream now?

After this first solution, module "M" requires 8ms per iteration, for a total of 80ms for each array, instead of 200ms. It still is a bottleneck, but the completion time now is 80ms*100inputs=8000ms=8seconds, which is a good improvement.

TASK #4

What are the ideal performance of map and farm?

The farm and map paradigm ideal performances should allow the module "M" to compute one input array in a time that is closer to 0ms. In that scenario, "M" can perform all the operations over an array immediately. This way, the system would take only the time that the module "IN" needs to create the inputs.