Assignment 1

Foundations of HPC 2021-2022

Francesco Ortu | francescortu@live.it

Section 1

Ring

In order to solve the exercise I provide two different implementation: one it uses only blocking operations while the other one uses non-blocking operations.

Implementation

In the blocking implementation I have divided the cores in two subset based on the parity of ranks. The odd processors send to right and left, and after they receive from left and right. Instead, the even processors first receive from left and right and after send to right and left. In this way the program avoid deadlock. Obviously, when the cardinality of the processors is an odd number I had to modify the behaviour of the processor with rank 0. In this case, the processor 0 send to left and receive from right, and after send to right and receive from left. The execution in the two cases is shown in figure 1.

The implementation that use only non-blocking operations is simpler: each processor send and receive from e to its neighbours. Then, there is a barrier in order to prevent a processor from update its message before it has received it. This barrier implicitly makes the execution "blocking", however as seen in the next section, this implementation is slightly faster then the other one.

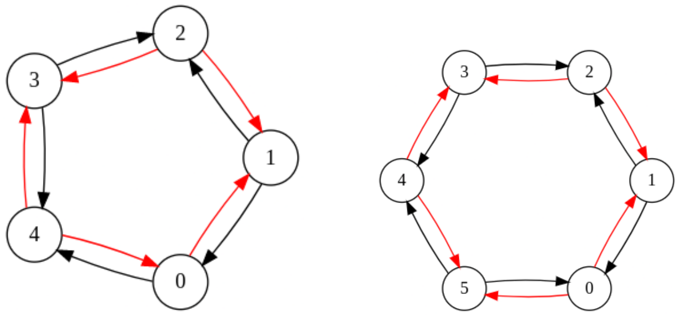

Figure 1: The blocking implementation with odd and even numbers of processors. In black there are the first operation executed while in red the second one.

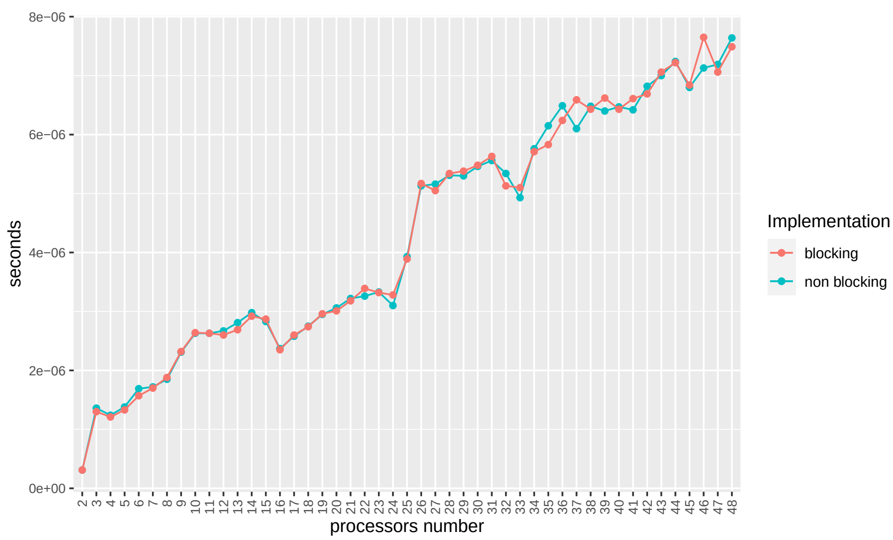

Figure 2: Walltime of the ring as a function of the number of processors

Runtime analysis

Since each processor send and receive $4n$ message, where n is the total number of processors, the runtime is expected to grow linearly on n. Meanwhile, the total number of messages exchanged between processors grows as $O(n^2)$.

The runtime is calculated as the time of the slowest core. In addition, to obtain significant data, I have taken the mean over 100000 repetition. The figure 3 show the time as a function of the number of processors used, i.e the number of vertices of the ring. As we expected, the time grows linearly. Even if there is some noisy, we can notice the existence of a great step between 23 and 24 processors due to the fact of using two nodes.

Matrix-Matrix sum

MAP	NProc	Total.Time	Jacobi.Time	Comm.Time	MLUPs	k	Latency[usec]	Band[MB/s]	C(L,N)[Mb]	T_s[s]	Tc(L,N)[s]	P(L,N)[MLUPs]	s]
core	1	15.2349570	15.0481740	0.18678295	113.4234	0	0.0000	0.00	0.00000	15.2349570	NA	NA	
core	4	3.8313109	3.7645463	0.06013802	451.0205	4	0.2469	20845.50	36.57372	3.8087392	0.0002193142	453.6673	
core	8	1.9413707	1.8877412	0.04911357	890.0923	6	0.2469	20845.50	34.56000	1.9043696	0.0002072390	907.2881	
core	12	1.3053588	1.2642190	0.03861560	1323.7728	6	0.2469	20845.50	26.37422	1.2695797	0.0001581529	1360.9108	
socket	4	3.8276910	3.7630817	0.05929418	451.4470	4	0.6231	22748.60	36.57372	3.8087392	0.0002009669	453.6695	
socket	8	1.9366420	1.8854327	0.04773249	892.2656	6	0.6231	22748.60	34.56000	1.9043696	0.0001899018	907.2964	
socket	12	1.2938464	1.2572978	0.03453514	1335.5507	6	0.6231	22748.60	26.37422	1.2695797	0.0001449222	1360.9250	
node	12	1.2963688	1.2581940	0.03474836	1332.9503	6	1.1328	11914.97	26.37422	1.2695797	0.0002766920	1360.7837	
node	24	0.6545812	0.6306265	0.02099294	2639.8432	6	1.1328	11914.97	16.61472	0.6347899	0.0001743050	2721.4134	
node	36	0.4439099	0.4213099	0.02070858	3892.6559	12	1.1328	11914.97	25.35880	0.4231932	0.0002660393	4080.6756	
node	48	0.3404002	0.3191919	0.01622802	5076.3440	12	1.1328	11914.97	20.93323	0.3173949	0.0002196106	5440.5568	

Figure 3: Walltime of the ring as a function of the number of processors

Section 2

OpenMPI

IntelMPI

Section 3