Multicore/Multinode hands-on exercises

Francesco Cicala

Assignment 5

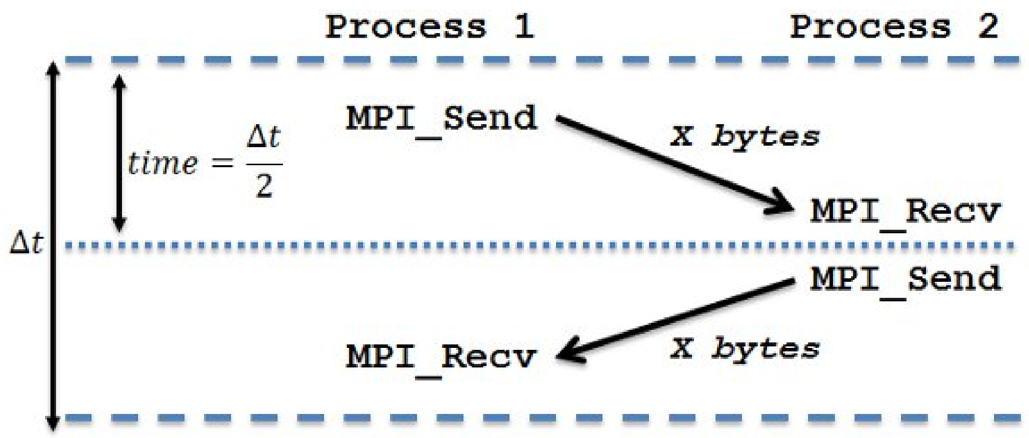
Abstract

In this assignment we have tested internode, intercore performances on SISSA's Ulysses cluster. It has been done by means of some tools: the PingPong Intel's benchmark, the STREAM benchmark, hwloc, numactl and likwid. The full procedure is divided in three exercises: a measure of latency among MPI processes, bandwidth measures and the run of a program, nodeperf.c.

1. Latency and bandwidth measure with IMPI PingPong benchmark

The Intel MPI is a set of benchmarks for performance measurements regarding communication operations which are capable to characterize performances of a cluster system, network latency, throughput and the efficiency of an MPI implementation.

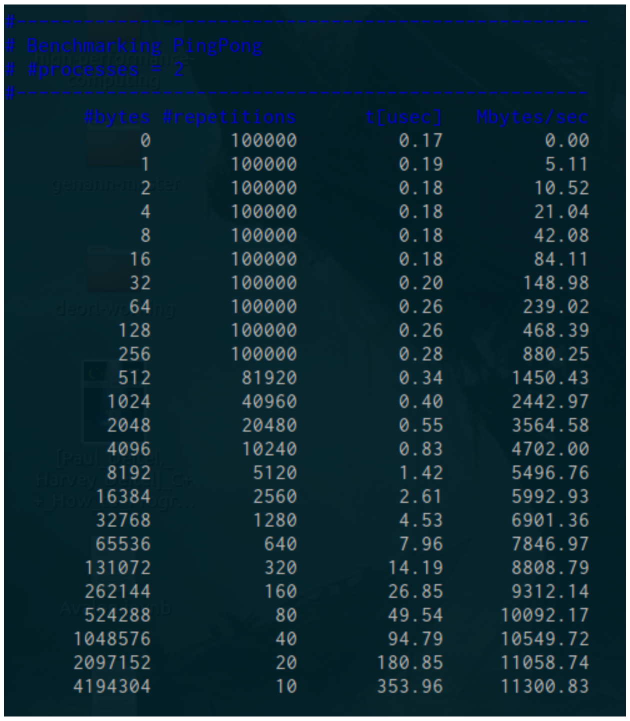
We have used the PingPong benchmark to assess the latency and the bandwidth during the exchange of packets with in a range of different sizes.



We show the results for two cases: intrasocket and intranode (intersocket).

1.1. Intrasocket

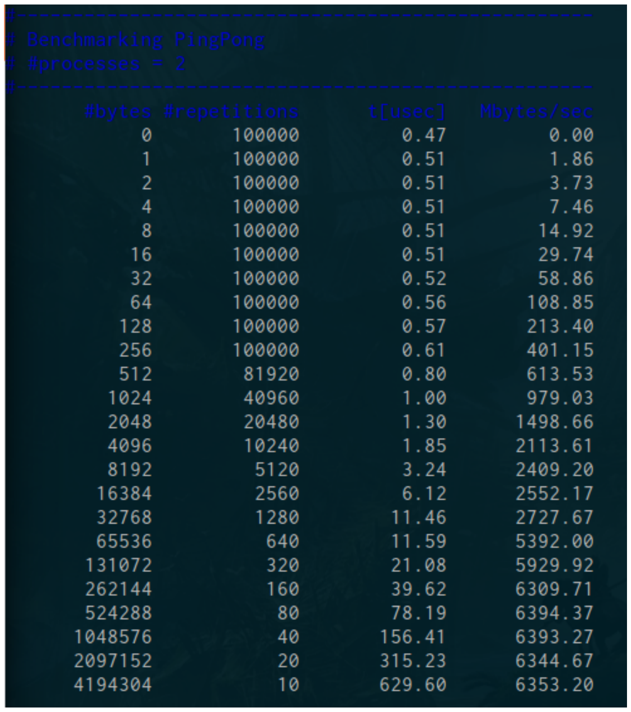
```
mpirun −np 2 hwloc−bind core:0 core:1 \
/u/shared/programs/x86_64/intel/impi_5.0.1/bin64/IMB−MPI1 \
PingPong −iter 100000 > run.dat
```



```
#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0       100000         0.17         0.00
            1       100000         0.19         5.11
            2       100000         0.18        10.52
            4       100000         0.18        21.04
            8       100000         0.18        42.08
           16       100000         0.18        84.11
           32       100000         0.20       148.98
           64       100000         0.26       239.02
          128       100000         0.26       468.39
          256       100000         0.28       880.25
          512        81920         0.34      1450.43
         1024        40960         0.40      2442.97
         2048        20480         0.55      3564.58
         4096        10240         0.83      4702.00
         8192         5120         1.42      5496.76
        16384         2560         2.61      5992.93
        32768         1280         4.53      6901.36
        65536          640         7.96      7846.97
       131072          320        14.19      8808.79
       262144          160        26.85      9312.14
       524288           80        49.54     10092.17
      1048576           40        94.79     10549.72
      2097152           20       180.85     11058.74
      4194304           10       353.96     11300.83
```

1.2. Intersockets

```
mpirun −np 2 hwloc−bind core:0 core:13 \
/u/shared/programs/x86_64/intel/impi_5.0.1/bin64/IMB−MPI1 \
PingPong −iter 100000 > run.dat
```



```
#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0       100000         0.47         0.00
            1       100000         0.51         1.86
            2       100000         0.51         3.73
            4       100000         0.51         7.46
            8       100000         0.51        14.92
           16       100000         0.51        29.74
           32       100000         0.52        58.86
           64       100000         0.56       108.85
          128       100000         0.57       213.40
          256       100000         0.61       401.15
          512        81920         0.80       613.53
         1024        40960         1.00       979.03
         2048        20480         1.30      1498.66
         4096        10240         1.85      2113.61
         8192         5120         3.24      2409.20
        16384         2560         6.12      2552.17
        32768         1280        11.46      2727.67
        65536          640        11.59      5392.00
       131072          320        21.08      5929.92
       262144          160        39.62      6309.71
       524288           80        78.19      6394.37
      1048576           40       156.41      6393.27
      2097152           20       315.23      6344.67
      4194304           10       629.60      6353.20
```

1.3. Comparison

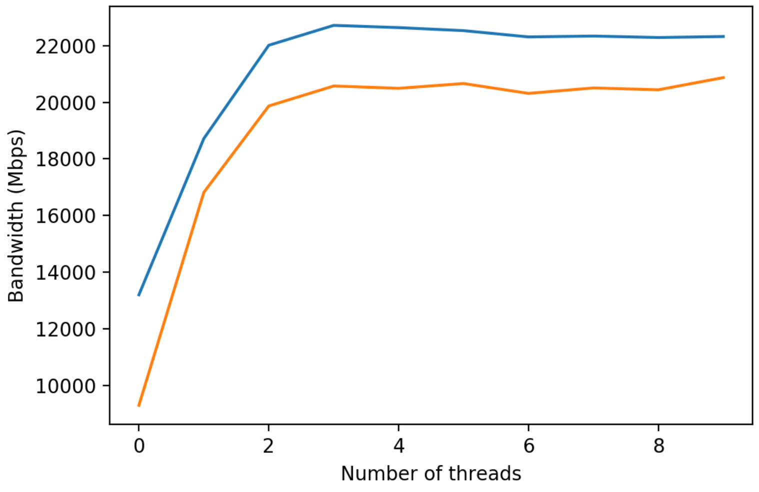
It is clear that optimizing code by being aware of the core on which each process is running can make a relevant difference in performances, when data exchange is involved. In fact, when the cores belong to the same socket, the latency is $0.17\mu s$ and bandwidth reaches 11 Gbps by increasing the size of the packets. On the other hand, the intersockets case shows a significant increase in latency, $0.47\mu s$, and a bandwidth which is nearly a half of the previous one, approximately 6 Gbps.

2. Stream benchmark

Stream is a synthetic benchmark program that measures sustainable memory bandwidth. We have compiled it and executed the benchmark by means of the command:

```
(for i in `seq 1 10`; do OMP_NUM_THREADS=$i numactl --cpunodebind 0
--membind 0 ./stream.x; done) | grep "Triad:" | awk '{print $2}'
```

The second time, we have changed the value of membind to 1. In this way we have two measurements, each composed of ten different bandwidth measures (MB/s) for the number of threads in the range from 1 to 10.



We observe that, for the same number of threads, by using the local memory the performance is higher. This is easily explainable by considering that if the memory is local the data runs over shorter paths.

3. Nodeperf

We have firstly compiled the nodeperf.c file by using the Intel compiler, and then by means of the GNU compiler. The output is an executable, which computes matrix-matrix multiplication in order to measure the peak performance of an Ulysses node.

The command we used to invoke the Intel compiler is:

```
mpiicc −O2 −xHost −fopenmp −mkl nodeperf.c −o nodeperf.x
```

On the other hand, to compile it with the GNU compiler:

```
mpicc −fopenmp −o3 nodeperf.c −m64 −I${MKLROOT}/include −o
nodeperf.x −L${MKLROOT}/lib/intel64 −Wl,−−no−as−needed
−lmkl_intel_lp64 −lmkl_sequential −lmkl_core −lpthread −lm −ldl
```

The executable produced by the Intel compiler has a performance of 442 GFlops, while the other one reached 27 GFlops.

The peak performance of a node is 48 GFlops. Therefore, the first executable reached the 98,6%, while the second one only the 6%. We conclude that the highly optimized compiler completely surpass the GNU one, and so, in general, the choice of the compiler can be hugely relevant in a performance perspective.