IMPI Benchmark (ping-pong) to measure latency among MPI processes assigned on different cores

Nicole Orzan

December 21, 2017

The aim of today's exercise was to run intel MPI ping-pong benchmark among to processors within a node and try to estimate latency and bandwidth. I worked on node cn07-33.
To execute the IMPI benchmark, first of all I had to load the following module:

```
module load impi-trial/5.0.1.035
```

There are 2 ways we can measure the latency: inside the same socket or between two different sockets. Do do this we can use the "hwloc" command, which permits to bind a processes to a given CPU set.
To measure the latency inside the same socket we have to specify the numbers of 2 cores from the same socket, while to measure it between two sockets we have to specify the numbers of 2 cores from different sockets.
To know which cores are in socket 0 and which are in socket 1 we can use the command numactl –hardware. In this case I obtained the result:

```
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9
node 0 size: 163811 MB
node 0 free: 160034 MB
node 1 cpus: 10 11 12 13 14 15 16 17 18 19
node 1 size: 163840 MB
node 1 free: 159807 MB
node distances:
node   0   1
  0:  10  11
  1:  11  10
```

So for the latency inside socket 0 I used the command:

```
mpirun -np 2 hwloc-bind core:0 core:5 /u/shared/programs/x86_64/
intel/impi_5.0.1/bin64/IMB-MPI1 PingPong
```

While for the latency between two sockets I had to change:

```
hwloc-bind core:0 core:13
```

We can also specify the number of iterations to use writing "-iter num". I decided to use the default number of iterations (num=1000) and to perform 3 runs for each of the two cases.
For the first case (cores 0-5), the result I obtained for # bytes from 0 to 16 are the following:

```
       #bytes #repetitions      t[usec]   Mbytes/sec
            0         1000         0.18         0.00
            1         1000         0.19         5.05
            2         1000         0.20         9.42
            4         1000         0.20        18.83
            8         1000         0.18        41.34
           16         1000         0.18        82.69

            0         1000         0.21         0.00
            1         1000         0.22         4.40
```

```
          2       1000       0.23       8.44
          4       1000       0.20      18.62
          8       1000       0.21      36.59
         16       1000       0.22      69.19

          0       1000       0.21       0.00
          1       1000       0.22       4.27
          2       1000       0.23       8.46
          4       1000       0.22      17.62
          8       1000       0.22      35.32
         16       1000       0.23      67.51
```

For the second case (cores 0-13) instead, the results obtained are:

```
     #bytes #repetitions    t[usec] Mbytes/sec
          0       1000       0.57       0.00
          1       1000       0.65       1.47
          2       1000       0.67       2.84
          4       1000       0.66       5.78
          8       1000       0.66      11.50
         16       1000       0.65      23.31

             0       1000       0.59       0.00
          1       1000       0.63       1.50
          2       1000       0.63       3.04
          4       1000       0.62       6.14
          8       1000       0.62      12.39
         16       1000       0.62      24.46

          0       1000       0.56       0.00
          1       1000       0.64       1.48
          2       1000       0.63       3.02
          4       1000       0.64       5.94
          8       1000       0.63      12.05
         16       1000       0.63      24.41
```

We can clearly say that the latency is higher when we bind the process to work on 2 different sockets. Calculating a mean from the outcomes obtained (as the average of all the 18 values, for the 2 different cases), we can say that in the case in which we have only one socket the value of the latency is 0.20 ± 0.01 Mbytes/sec, while in the other case its value is 0.62 ± 0.02 Mbytes/sec.