Multicore and Multinode Performance

Gabriele Sarti

December 5, 2018

Abstract

This assignment is divided into three hands-on exercises which are intended to measure the performances of Ulysses for multicore and multinode, intercore and intracore computations through the use of I/O benchmarks.

1 Introduction

PingPong is a simple benchmark created by Intel for measuring latency and throughput of a single message sent between two processes. [1] The source of the message can be specified in order to evaluate latency between processors which are on different sockets, or even inside different nodes. The size of the message sent is increased progressively in order to estimate system's maximal bandwidth.

The Sustainable Memory Bandwidth Benchmark (STREAM) is a simple synthetic benchmark that measures sustainable memory bandwidth (in MB/s) and the corresponding computation rate for four simple vector kernels, namely the copy, scale, add and triad operations. It was created in 1991 by Professor McCalpin, who also published an article about it in 1995 [2].

Using STREAM to evaluate the performances of an HPC system has become a mainstream practice since performance limitations of a modern system are mostly caused by memory bandwidth. The code needed to compile and run the benchmark is available on Virginia University website. [3]

Latency for the HPC context is defined as the delay in communication between different elements of a cluster [4]. Bandwidth is defined as the maximum throughput of a communication channel between two components of a digital system. [5]

2 Setup Procedure

2.1 PingPong Setup

The compiled executable is already available inside the impi-trial module of Ulysses cluster. We execute the following commands in order to measure latency and bandwidth of our

system

```
# Load the benchmark
module load impi-trial/5.0.1.035
# Set a variable for the PingPong path
PingPongPath=/u/shared/programs/x86_64/intel/impi_5.0.1/bin64/IMB-MPI1
```

Listing 1: Measure latency and bandwidth for MPI communications on Ulysses

2.2 STREAM Setup

We created a bash script in order to compare bandwidth for reading close and distant memory for all threads available in one socket of Ulysses. For each run we computed the bandwidth for close and distant reads twice, swapping the cores used to read and the socket to which the memory was associated.

The run_stream.sh file is available in Appendix A.

2.3 Nodeperf Setup

In order to make the nodeperf.c benchmark functional, we replace mkl_malloc with a normal malloc and replace the argument -qopenmp used in the Assignment description with -fopenmp.

3 Execution

3.1 PingPong Execution

We use the following execution parameters for running the PingPong benchmark.

```
------------------------------------------------------------
    Intel (R) MPI Benchmarks 4.0, MPI-1 part
------------------------------------------------------------
Machine                : x86_64
System                 : Linux
Release                : 2.6.32-573.12.1.el6.x86_64
Version                : #1 SMP Wed Dec 16 11:39:16 CET 2015
MPI Version            : 3.0
Node                   : cn08-17
Number of Processes    : 2
```

We run the following commands in order to measure latency, bandwidth and estimate default intranode and internode communication performances.

```
# General execution with one million iterations.
mpirun -np 2 $PingPongPath -iter 1000000 PingPong
# Close intranode execution, gives latency between same-socket cores.
```

```
mpirun -np 2 hwloc-bind core:0 core:7 $PingPongPath -iter 1000000 PingPong
# Distant intranode execution, gives latency between different-socket cores.
mpirun -np 2 hwloc-bind core:0 core:13 $PingPongPath -iter 1000000 PingPong
# Internode execution, gives latency between different-node cores.
# -npernode was deprecated in favor of -map-by ppr:n:node
mpirun -np 2 -map-by ppr:1:node $PingPongPath -iter 1000000 PingPong
```

Our results for the PingPong benchmark are available in the attachment pingpong.txt. We present a summary of those results in the two subsections below.

3.1.1 Latency and Bandwidth Estimation

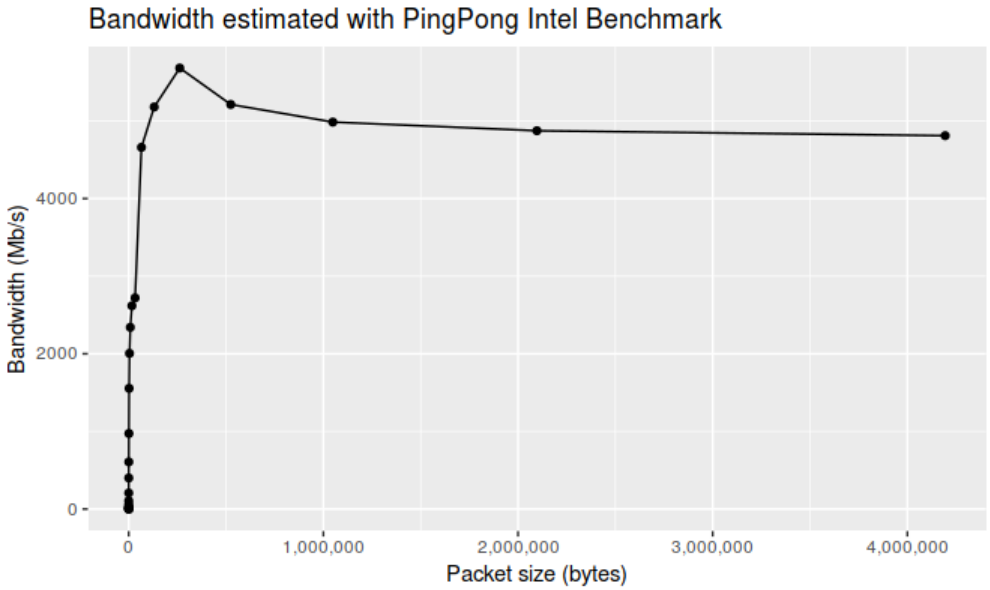
Table 1 contains the average values computed on three non-parametrized executions of the benchmark for one million iterations each.

Table 1: Results of the PingPong Benchmark. Tot. Rep. represent the total number of repetitions, $\bar{T}$ is the average time of execution and $\bar{B}$ is the average bandwidth. Single values for the bandwidth were computed as $B = \frac{\#\text{ bytes}}{T}$

# bytes	Tot. Rep.	$\bar{T}$ (usec)	$\bar{B}$ (Mb/s)
0	3000000	0.48	0.00
1	3000000	0.52	1.83
2	3000000	0.52	3.66
4	3000000	0.52	7.32
8	3000000	0.52	14.61
16	3000000	0.52	29.21
32	3000000	0.53	56.80
64	1966080	0.58	105.54
128	983040	0.59	206.15
256	491520	0.61	400.22
512	245760	0.80	607.17
1024	122880	1.00	973.48
2048	61440	1.25	1555.06
4096	30720	1.95	2005.56
8192	15360	3.34	2341.16
16384	7680	5.94	2618.89
32768	3840	11.18	2720.48
65536	1920	13.37	4660.08
131072	960	24.12	5180.55
262144	480	44.07	5681.79
524288	240	94.52	5212.04
1048576	120	199.69	4985.36
2097152	60	410.32	4874.20
4194304	30	833.50	4810.39

From the definition presented in the introduction, we can deduce that latency is the average time of communication for almost-empty packets, which in this case is approximately 0.52 usecs for packet sizes ranging from 0 to 32 bytes. From Figure 1 we can see that the values of bandwidth tend to flatten for large packets. We can estimate bandwidth at around 5000 Mb/s for this Ulysses' node.

Figure 1: Bandwidth obtained by passing different packet sizes to a non-parametrized Intel PingPong benchmark.



3.1.2 Gauging Defualt Intranode/Internode Communication Performance

After executing the parametrized PingPong benchmark through the commands mentioned at the beginning of this section for three times each, we obtained an estimate of the communication performances of the system inside and between sockets and nodes.

It is important to note that, in order to perform an intranode execution on the Ulysses cluster, we had to recompile the IMB-MPI1 file containing the PingPong benchmark from the original source [6], and run it using the OpenMPI implementation of mpirun instead of the IMPI one, which wasn't included in the guidelines for the assignment.

Figures 2 and 3 were created using data from those executions, and clearly emphasize the theoretical assumptions we made before performing those tests: the increase in distance between components translates in a degradation of communication performances including both latency and bandwidth. This behavior is normal, given the fact that Ulysses processors implement the NUMA architecture.

Figure 2: Bandwidths of different memory access conformations estimated with Intel Ping-Pong benchmark.

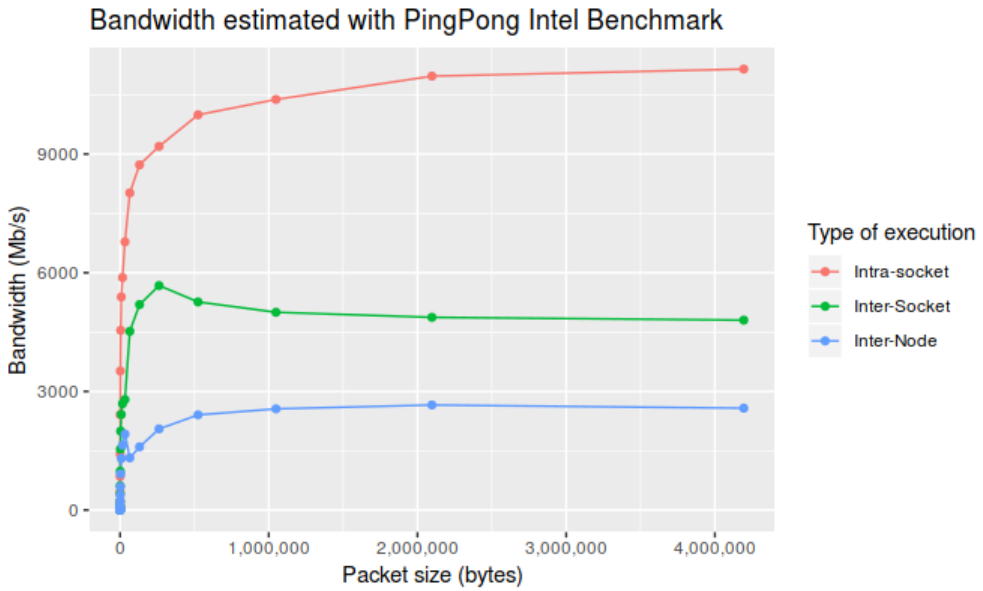
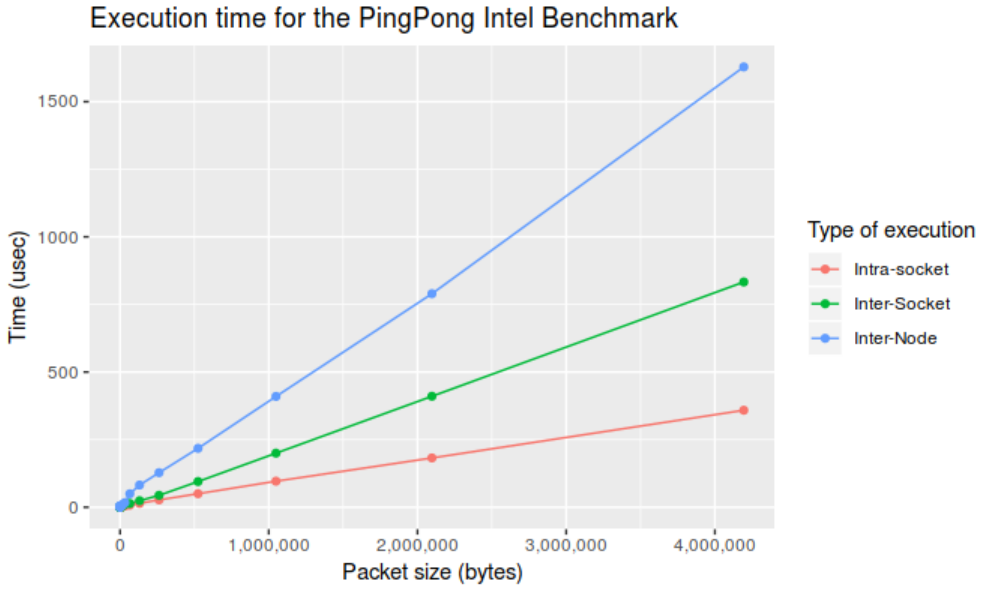


Figure 3: Execution time for different memory access conformations with the Intedl Ping-Pong benchmark.



3.2 STREAM Execution

To estimate the bandwidth of a whole Ulysses node, we run the stream_omp executable without any parameter using the numactl command in interactive mode. The program automatically allocated the maximum number of threads, which is 20 in our case, and returned the following results:

```
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           30196.6     0.042856     0.042389     0.043792
Scale:          29421.9     0.043664     0.043505     0.043880
Add:            34367.4     0.056231     0.055867     0.058665
Triad:          34712.1     0.055712     0.055312     0.060525
```

In order to estimate the bandwidth of a single core, we changed the value for the global variable OMP_NUM_THREADS from the default to one before executing again the same command, obtaining the following performances:

```
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           13133.2     0.097537     0.097463     0.097820
Scale:          12982.5     0.098705     0.098594     0.099905
Add:            13662.9     0.140651     0.140526     0.140963
Triad:          13661.8     0.140685     0.140538     0.141012
```

Then we proceeded in computing the bandwidths for one single cores reading from memory associated to their own socket against those of the same cores reading from distant memory, using the script run_stream.sh available in Appendix A. The results of its execution are available in Attachment stream.txt.

Figure 4: Bandwidths of different memory access conformations obtained with STREAM benchmark.

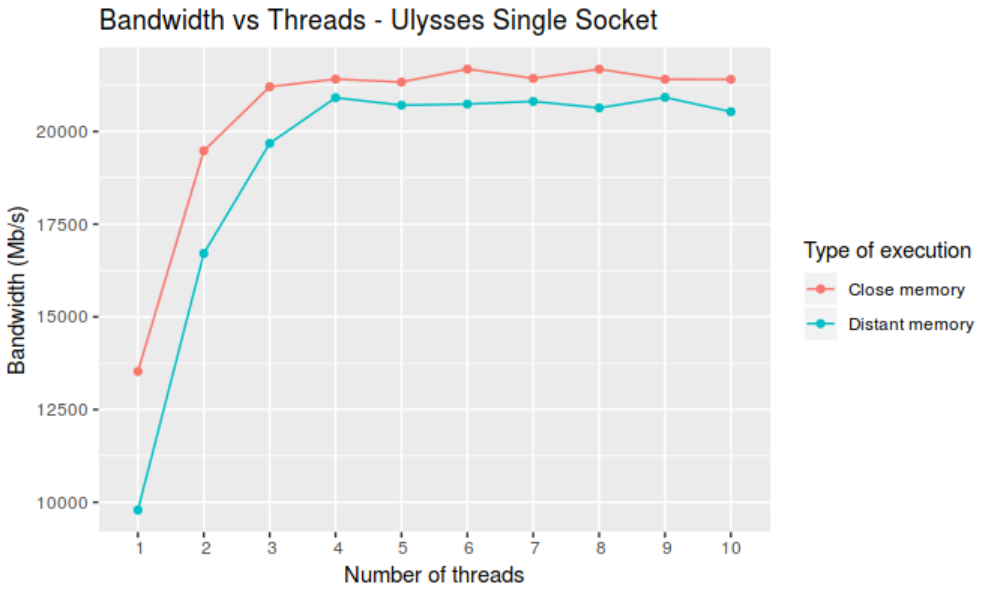


Figure 4 provide a graphical representation of our results, with each point being the best bandwidth obtained for 50 iterations using the respective number of threads on the Triad operation. It is evident how the original bandwidth varies heavily with respect to close or distant memory, but this difference becomes almost negligible when the communication is performed by a large number of threads.

3.3 Nodeperf Execution

In order to obtain an executable we loaded the modules as follows and we compiled the nodeperf.c file contained in the assignment, obtaining the following result

```
Multi-threaded MPI detected

This driver was compiled with:
  -DITER=4 -DLINUX -DNOACCUR -DPREC=double
Malloc done.  Used 1846080096 bytes
NN lda=15000 ldb= 192 ldc=15000 0 0 0 438456.082 cn08-17
```

The performance obtained is 438.46 GFlops, which is approximately 97.87% of peak performance. We also tried to perform a run without using Intel compiler, replacing the imalloc routine with a normal malloc.

```
Multi-threaded MPI detected

The time/date of the run...  at Wed Nov 28 22:16:10 2018

This driver was compiled with:
  -DITER=4 -DLINUX -DNOACCUR -DPREC=double
Malloc done.  Used 1846080096 bytes
(0 of 1): NN lda=15000 ldb= 192 ldc=15000 0 0 0 26986.257 cn08-17
```

We can see that in this case we obtain 26 GFlops as result, a performance which accounts only for 6% of theoretical peak performance for a node.

References

[1] `https://software.intel.com/en-us/imb-user-guide`

[2] Sustainable memory bandwidth in current high performance computers, JD McCalpin - Silicon Graphics Inc, 1995

[3] `https://www.cs.virginia.edu/stream/FTP/Code/`

[4] `https://en.wikipedia.org/wiki/Latency_(engineering)`

[5] `https://en.wikipedia.org/wiki/Bandwidth_(computing)`

[6] We cloned this repository `https://github.com/intel/mpi-benchmarks` and compiled the code using the Makefile, following the instructions given in the READMEs.

Appendices

A run_stream.sh

```
# Executing the STREAM benchmark with multiple num threads

# Print execution node name

/bin/hostname

# Enter the right directory

cd ~/high_performance_computing/AssignmentsDSSC/A5-multicore-multinode/stream/

# Compile the code
make

# Load the openmpi module
module load openmpi/1.8.3/gnu

#Run the STREAM benchmark with the specifications
for threads in 1 2 3 4 5 6 7 8 9 10
do
  export OMP_NUM_THREADS=${threads}
  echo "---------------------"
        echo "Run ${runs} with ${threads} thread(s)"
        echo "---------------------"
        echo "**** CLOSE:"
        numactl --cpunodebind=0 --membind=0 ./stream_omp.x
        echo "**** DISTANT:"
        numactl --cpunodebind=0 --membind=1 ./stream_omp.x
        echo "****REVERSED CLOSE:"
        numactl --cpunodebind=1 --membind=1 ./stream_omp.x
        echo "****REVERSED DISTANT:"
        numactl --cpunodebind=1 --membind=0 ./stream_omp.x
done

exit
```