Multinode / Multicore hands-on exercises

IMPI benchmark for bandwidth

The IMPI benchmark ("PingPong") is a C program exploiting MPI's message passing to measure communication efficiency (bandwidth) between two or more cores located in a cluster. The bandwidth is recorded for messages of several sizes (up to 2^{22} bytes).

As in the readme for the exercise, I ran the IMPI benchmark on two nodes on Ulysses (cn01-02 and cn01-06) using different configurations:

1. Computation and memory on cores within the same socket of the same node cn01-02
2. Computation and memory on cores within two different sockets of the same node cn01-02
3. Compitation and memory on two different nodes (cn01-02 vs cn01-06)

Each configuration was ran for three times.

This is the script used to run the program with such configurations:

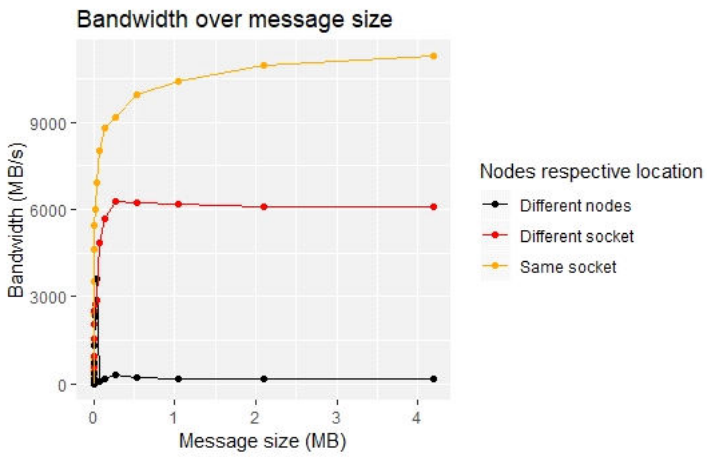
```
module load impi-trial/5.0.1.035

rm /home/mzullich/HPC/03-multinode_multicore/pingpong_hwloc.txt

for i in $(seq 1 3); do
     mpirun -np 2 hwloc-bind core:0 core:7
/u/shared/programs/x86_64/intel/impi_5.0.1/bin64/IMB-MPI1 PingPong -iter
1000000 | egrep '^[ ]+[0-9]' | sed -r 's/^/same_socket/g' >>
/home/mzullich/HPC/03-multinode_multicore/pingpong_hwloc.txt

     mpirun -np 2 hwloc-bind core:0 core:13
/u/shared/programs/x86_64/intel/impi_5.0.1/bin64/IMB-MPI1 PingPong -iter
1000000 | egrep '^[ ]+[0-9]' | sed -r 's/^/diff_socket/g' >>
/home/mzullich/HPC/03-multinode_multicore/pingpong_hwloc.txt

     mpirun -np 2 -map-by ppr:1:node
/u/shared/programs/x86_64/intel/impi_5.0.1/bin64/IMB-MPI1 PingPong -iter
1000000 | egrep '^[ ]+[0-9]' | sed -r 's/^/diff__nodes/g' >>
/home/mzullich/HPC/03-multinode_multicore/pingpong_hwloc.txt
done

module purge
```

The script was ran specifying the nodes' names in the request:

```
qsub -l walltime=3600,nodes=cn01-02+cn01-06 myscript.sh
```

NB: the inter-node message passing was run with the -map-by flag since -npernode is deprecated within the current MPI module.

I recorded the worst time for each repetition and message size; these are the results I got for the bandwidth:



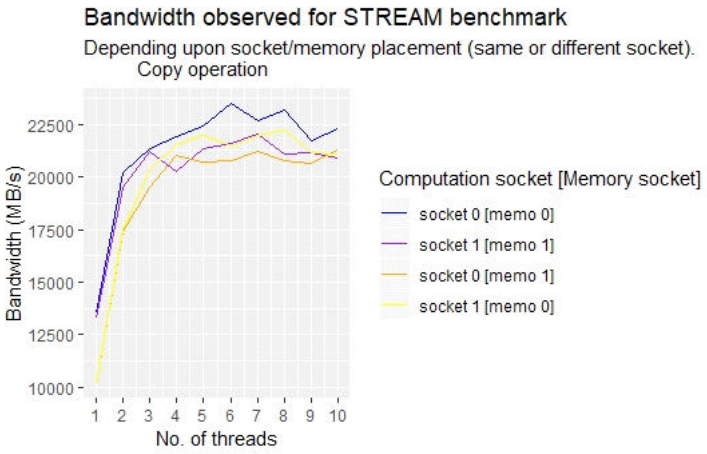
We see how:

- As expected, the intra-socket communication is the fastest one, as the bandwidth recorded is over 10 GB/s for message of size over 2 MB;
- The inter-socket, inter-node communication is well slower, as the recorded bandwidth quickly settles at around 6 GB/s (almost half of the intra-socket case) for messages of a few KB;
- The inter-node communication instead shows a weird behaviour: after being in-line with the inter-socket run, at ~3KB of message size, after reaching a bandwidth of ~3.6 GB/s, it plummets to an abysmal rate of ~150 MB/s for larger message sizes. I can't give and answer to this behaviour.

STREAM benchmark

The STREAM benchmark is another benchmark (using OMP paradigm) for recording bandwidth for four different tasks (copy, scale, add, triad). As in "PingPong", the user may select where to allocate memory and computation.

I executed four runs while using different configurations of computation vs memory socket, varying between the two socket of node cn01-02, while varying the number of processors involved in the computation:



With all configurations, the bandwidths show a plateau after 5 threads. The best result was for computation and memory on socket 0 (peak at 6 threads, ~24 GB/s bw), while the worst was, as expected, for memory and computation on different sockets (namely, 0 and 1 respectively). The same configuration but with inverted sockets gives better results, although not good as in the 0-0 case; eventually, the 1-1 configuration shows mixed results: at first it keeps up with the omologous 0-0, but then it shows a mixed behaviour, very similar to the other two (inter-socket): to stabilize the results, maybe a better idea would be to repeat the experiments and average the bandwidth over multiple runs, as the measurements don't seem to be as stable as in the IMPI benchmark.

Nodeperf

Nodeperf benchmark measures nodes' performance in term of Flops.

In order to compile it on Ulysses, the `imalloc` routine within `nodeperf.c` was modified such that the `mkl_malloc` is replaced by a regular `malloc`.

This is the script used to compile, run and store results:

```
#module to import

module purge
module load openmpi
module load intel
module load mkl
module load impi-trial/5.0.1.035 #has mpiicc command

#compile with intel compiler

mpiicc -O2 -xHost -qopenmp -mkl nodeperf.c -o nodeperf_MPIICC

#compile with gcc

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

#set env variables

export OMP_NUM_THREADS=20

export OMP_PLACES=cores
```

```
#run both files & store results

./nodeperf_MPIICC > results_INTEL.txt
./nodeperf_GCC > results_GCC.txt
#cleanup modules

module purge
```

These are the results (for node `cn04-13`):

- INTEL-compiled executable:

```
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 446835.430 cn04-13
```

- GCC-compiled executable:

```
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 26998.839 cn04-13
```

The performance is indicated in bold underlined and is expressed in MFlops.

The theoretical peak performance for the node is 448000 MFlops (20 cores • 2.8 GHz • 8 flop/s). Given this data, the INTEL-compiled executable achieved 99.74% of the theoretical peak performance, while the GCC-compiled only 6.05%.