Multicore-Multinode

Inspecting Numanodes and threads placement

Rabindra Khadka

DSSC, UNITS

Overview:

Modern processors mostly have NUMA architecture. Multi socket computational nodes are significance in today's high performance computing world. This exercise was performed to investigate and understand NUMA and also to see the performance effect while migrating processes to different cores. Hyper threading was not enabled for this exercise. The exercise was divided into three sections on different topics such as 1. MPI 2. Stream and 3. Nodeperf. The results are explained under each topic.

Results and Analysis:

A: Stream:

Stream benchmark was taken from <https://www.cs.virginia.edu/stream/> to measure the sustainable memory bandwidth. The executable with the script seen in fig 1 below was ran with sequence of threads from 1 to 10 which executed stream_omp.x. While numactl binds all the threads to socket 0, the memory uses switches from within the socket to socket 1 in the second run .Similarly while the numactl binds all the threads to socket 1, the memory usage switches from within socket 0 to socket 1 in the third and fourth run respectively.

```
#!/bin/bash
for threads in `seq 1 10`;do
        export OMP_NUM_THREADS=${threads}
 numactl --cpunodebind=0 --membind=0 ./stream_omp.x>> stream_omp00.out
 numactl --cpunodebind=0 --membind=1 ./stream_omp.x>> stream_omp01.out
 numactl --cpunodebind=1 --membind=0 ./stream_omp.x>> stream_omp10.out
 numactl --cpunodebind=1 --membind=1 ./stream_omp.x>> stream_omp11.out

done
```

Fig 1: Shows the command to run numactl with different threads sequence

Number of Threads	BW (Cpu 0 -Mem 0)	BW (Cpu 0 -Mem 1)
1.0	13800.2	10071.6
2.0	20934.6	17635.2
3.0	23510.0	21669.2
4.0	24866.5	23085.3
5.0	25374.5	23129.0
6.0	25549.5	23331.5
7.0	25799.8	23291.5
8.0	25647.9	23567.3
9.0	25065.6	23366.8
10.0	25365.3	23009.1

Table 1: Bandwidth for memory bind within same socket and different socket

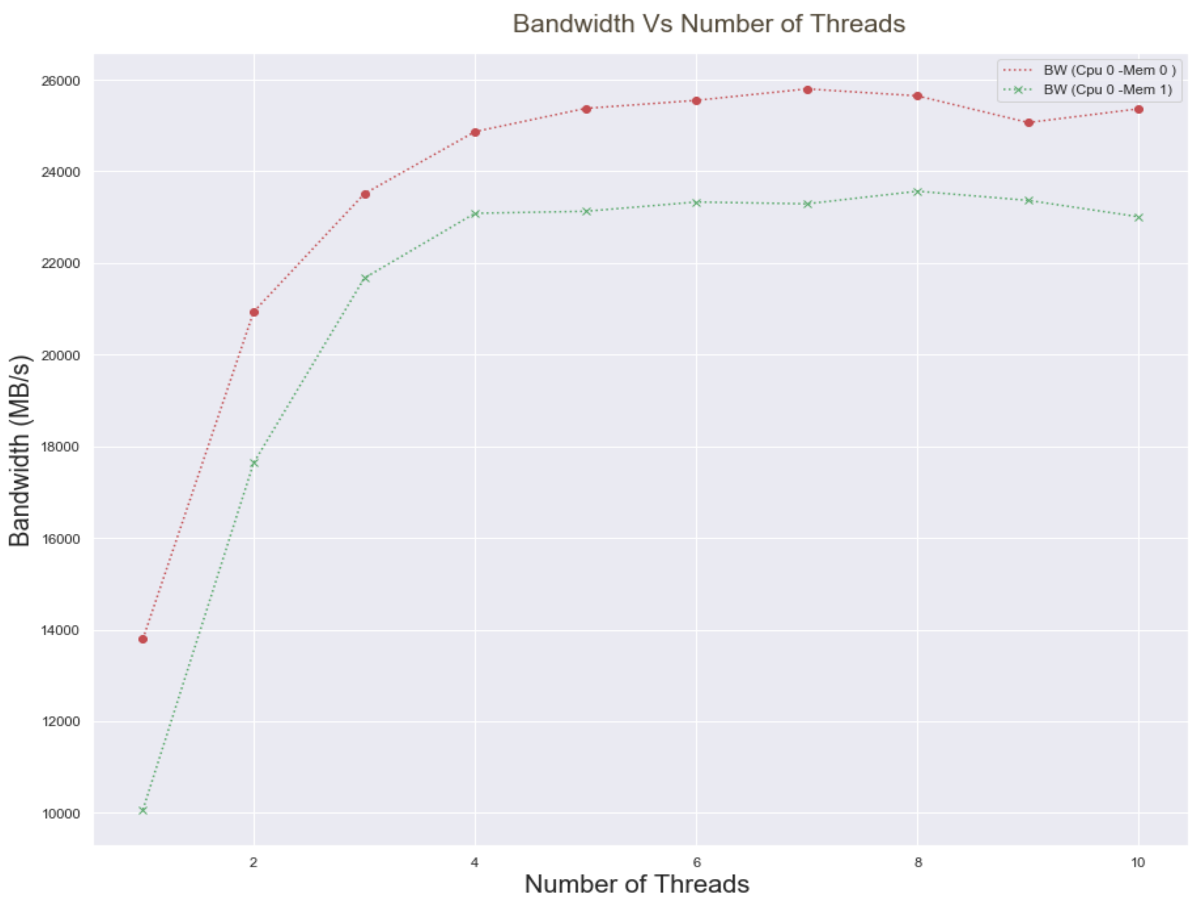


Fig 2: Comparison of BW between memory access within the socket and another socket

The above fig.2 depicts that the overall bandwidth of the Ulysses with memory access within the same socket is around 25 GB/s and for the single core it stands at approximately 14 GB/s. Similarly, we can read that while accessing memory from another socket the bandwidth shifts to around 10 GB/s.

B: MPI –Ping Pong

Intel MPI ping pong benchmark was executed to estimate the bandwidth and latency (module load impi – trial /5.0.1.035). Latency and Bandwidth were measured within the same sockets and between different sockets. The time taken to ping pong smallest messages were considered to estimate the latency (for almost no message packet) and for bandwidth estimation; bandwidth corresponding to larger messages were taken into consideration as bandwidth eventually stabilized.

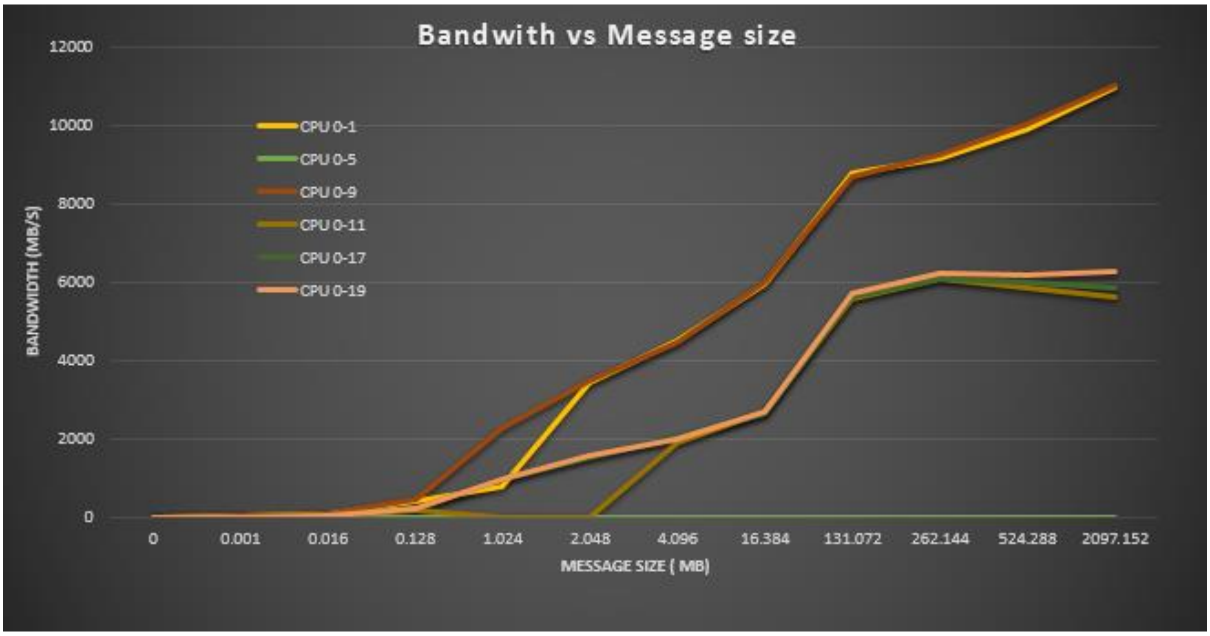


Fig 3: shows the bandwidth resulted from inter-socket ping and intra-socket ping within a node

In the above figure, it can be observed bandwidth forming two distinct clusters, the higher bandwidth clusters is resulted from the pinging among cores within the same socket and the lower bandwidth clusters resulted from the pinging between cores from different sockets within the same node. This segregation of bandwidth is explained by the fact that further the cores in term of distance, higher time it requires for the message packet to be exchanged. The bandwidth for Ulysses node can be approximated to be 5500 MB/sec.

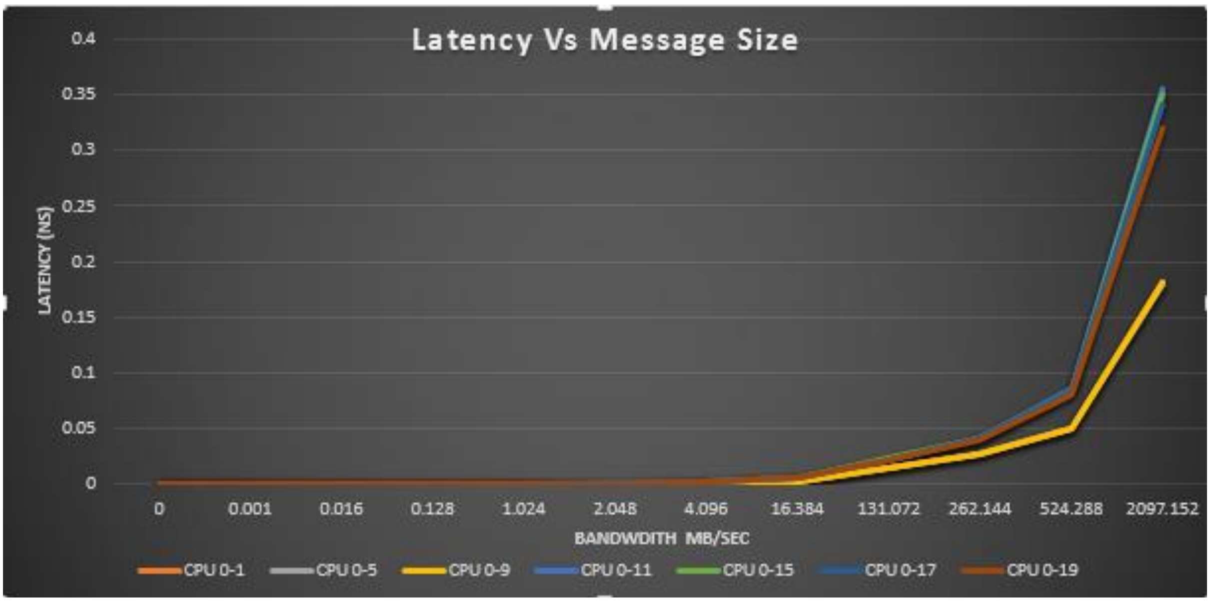


Fig 4: shows the latency resulted from inter-socket ping and intra-socket ping within a node

The figure 4 shows the segregation of latency into two clusters which can be explained by the fact that the inter socket pinging resulted in lower latency as shown by the yellow line while the intra-socket pinging resulted in higher latency as shown by the orange and blue curves. This highlights the significant of distances in determining the latency in a node. The Ulysses node latency can be approximated to be around 0.35 usec.

C: Intra-Node Ping Pong:

The intra node ping pong was performed to support our knowledge from the above internode ping pong which was concluded that the higher distance between the transmitter and receiver will result in increase in latency and decreases the bandwidth.

For performing the intra node ping pong following command was used:

```
mpirun -np 2 -ppn 20 -host node01,node02 ...
```

```
rkhadka@login2:~/Rabin/exercise5-multicpu-multicore/mpi/multinode
cn02-18
cn08-23
#------------------------------------------------------------
#    Intel(R) MPI Benchmarks 2019 Update 1, MPI-1 part
#------------------------------------------------------------
# Date                    : Fri Dec 14 10:22:16 2018
# 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
# MPI Thread Environment:
```

Fig 5: Two available nodes were cn02-18 and cn08-23 on which ping pong was

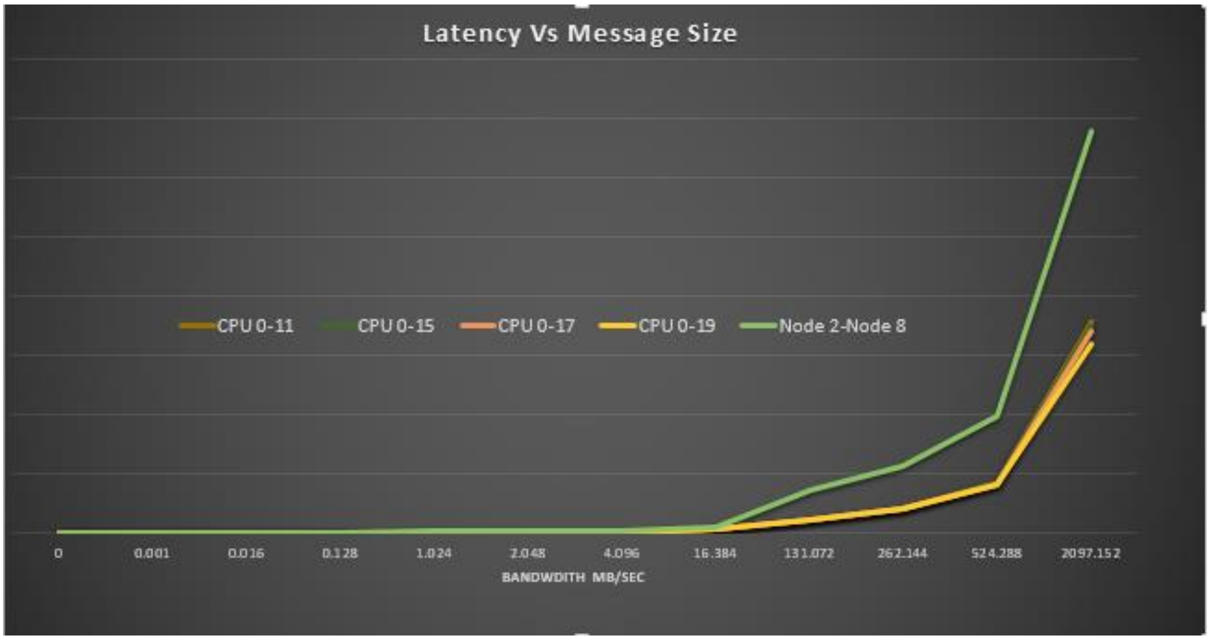


Fig 6: showing the latency when pinged inside the same node and between two different nodes.

The graph in fig 6 depicts the fact as the distance between the transmitter and receiver, in this case the cores increased the latency also increased. The green line shows the latency yielded when pinged between two different nodes and the clusters in yellow colour shows the latency when pinged within a single node.

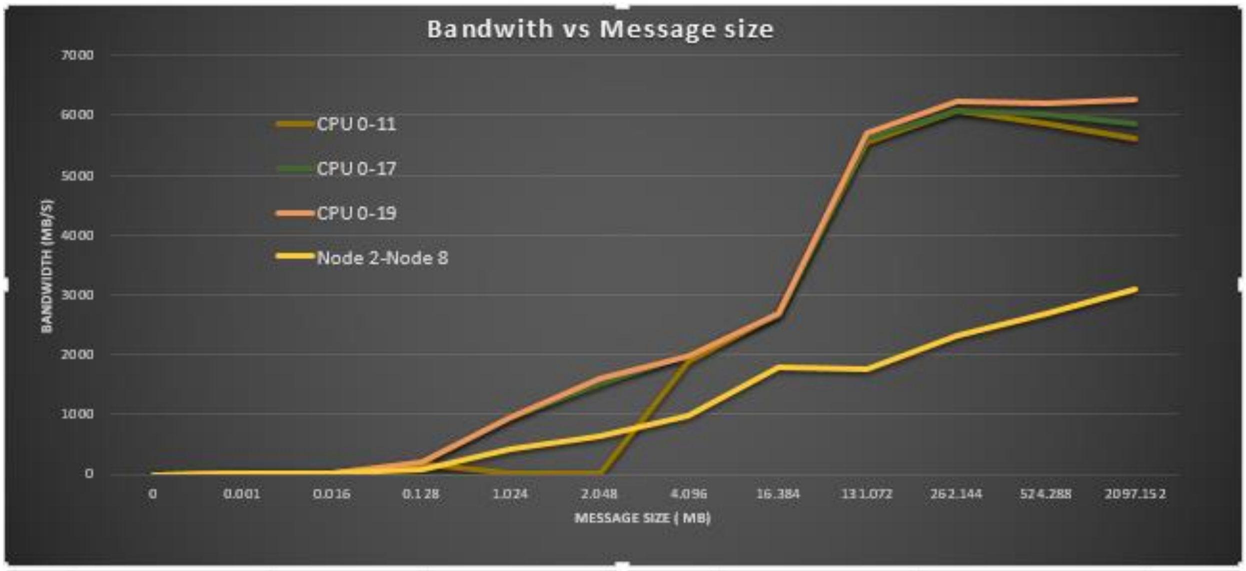


Fig7: showing the bandwidth with intra-nodal ping and inter-nodal ping of Ulysses cluster.

The bandwidth was also decreased when we had intra nodal ping between node 2 and node 8. This also supports the fact that distance communication within the nodes or a cluster suffers from the increase in latency and decrease in bandwidth.

C: Nodeperf

Nodeperf.c program was compiled using Intel MPI compile wrapper script and the peak performance of the single node in Ulysses was computed and compared. The threads of nodeperf.c program were simply distributed to 20 cores of a node in Ulysses and the peak performance was observed to be compared. In relation to this exercise, we replaced mkl_malloc with malloc and –qopenmp with –fopenmp.

```
$ mpiicc -O2 -xHost -qopenmp -mkl nodeperf.c -o nodeperf
```

OpenMP threads and their placements on the system was configured using:

```
$ export OMP_NUM_THREADS=20
$ export OMP_PLACES=cores
```

For execution;

```
./nodeperf.x
```

NodePerf (GFLOPS)	Theoritical Peak Performance (GFLPOS)
446.766977	20 cores × 2.8 GHz × 4 x 2 floats = 448 GF/s

Table 2: Comparing Nodeperf Gflops with theoretical peak performance

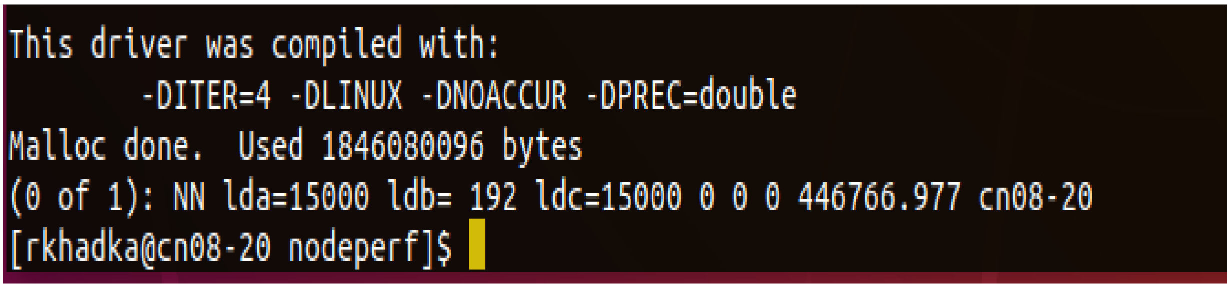
The Gflops yielded after running nodeperf is 446.76 Gflops which is at 99.7% of peak performance helped by multi-threading activation.

Summary:

The test performed in this lab exercise highlighted that the location of memory on multi socket platform will affect performance. By using command lines to bind processors with nearest memory could enhance the performance substantially.

The ping pong benchmark exercise showed that the communication distance plays a significant role in determining the performance of a hpc cluster. The placement of jobs between nodes or within a node must be carefully chosen by taking into account distance factor between nodes, memory and the cores. The higher the distances, larger the communication overhead.

APPENDIX A: Snippets of Nodeperf Result



```
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 446766.977 cn08-20
[rkhadka@cn08-20 nodeperf]$
```

APPENDIX B: Sample data of Ping Pong within a single node

Bandwidth(MB/sec)	Latency1(ns)	Bandwidth1(MB/sec)	Latency5(ns)	Bandwidth5(MB/sec)	Latency9(ns)	Bandwidth9(MB/sec)
0	0	0	0	0	0.0002	0
0.22	0.00021	4.47	0.0002	4.77	0.00021	4.46
3.44	0.00021	72.74	0.0002	78.21	0.00021	72.11
27.21	0.0003	405.62	0.00027	447.32	0.00028	434.58
214.52	0.00043	760.35	0.0004	2426.54	0.00043	2272.71
422.56	0.00057	3443.17	0.00054	3630.96	0.00056	3471.55
810.99	0.00087	4504.44	0.00083	4691.33	0.00087	4469.12
2672.64	0.00264	5924.16	0.00259	6041.38	0.00261	5988.01
190.95	0.01425	8772.86	0.01418	8818.28	0.01438	8691.96
371.91	0.02724	9178.41	0.02682	9321.97	0.02702	9253.84
256.36	0.05054	9892.23	0.04939	10124.14	0.04973	10055.27
206.37	0.18208	10984.17	0.17962	11134.34	0.18123	11035.83

Bandwidth11(MB/sec	Latency15(ns)	Bandwidth15(MB/sec)	Latency17(ns)	Bandwidth17(MB/sec)	Latency19(ns)	Bandwidth19(MB/sec)
0	0.00054	0	0.00055	0	0.00047	0
1.75	0.00061	1.55	0.0006	1.59	0.00052	1.84
28.25	0.00061	25.12	0.00063	24.34	0.00051	29.67
170.22	0.00062	196.9	0.00064	189.88	0.00057	212.89
0.94642	0.00104	938.37	0.00104	935.22	0.00102	961.71
1.54759	0.00125	1558.51	0.00194	1509.6	0.00123	1586.79
1891.19	0.00197	1985.04	0.00194	2011.81	0.00197	1985.99
2679.08	0.00596	2623.8	0.00589	2651.39	0.00581	2688.81
5560.16	0.02286	5467.12	0.0223	5605.02	0.02193	5698.78
6080.9	0.04113	6078.59	0.04105	6090.18	0.04015	6227.16
5862.57	0.08458	5911.53	0.08316	6012.8	0.08064	6200.58
5623.05	0.34972	5718.79	0.34207	5846.74	0.3197	6255.29

APPENDIX C: Sample data of Ping Pong between different nodes

Bandwidth(MB/sec)	latency (us)	Bandwith(MB/sec)
0	0.00163	0
0.22	0.00165	0.6
3.44	0.00179	8.92
27.21	0.00183	69.77
214.52	0.00252	406.76
422.56	0.00325	630.36
810.99	0.00417	981.31
2672.64	0.00909	1802.71
190.95	0.0741	1768.74
371.91	0.11367	2305.95
256.36	0.19609	2673.32
206.37	0.67809	3092.23