Stream Benchmark

Nicole Orzan

December 21, 2017

The aim of today's exercise session was to run the STREAM benchmark (taken from https://www.cs.virginia.edu/stream), which is a benchmark program that measures sustainable memory bandwidth (in MB/s) and the corresponding computation rate for simple vector kernels. We had to run it and to:

1. Estimate the bandwidth of one single core of ulysess;
2. Compare the bandwidth of two different cores, reading from memory associated to the socket of core 1 against the memory associated with the other socket;
3. Estimate the overall bandwidth of one node of ulysess;
4. Do it for all threads available in one socket and make a plot of the results..

I built stream.c using the provided Makefile, and I worked on Ulysses cluster, on the node cn08-05.

Output of the numactl - -show command:

```
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9
node 0 size: 81891 MB
node 0 free: 79785 MB
node 1 cpus: 10 11 12 13 14 15 16 17 18 19
node 1 size: 81920 MB
node 1 free: 79407 MB
node distances:
node   0   1
  0:  10  11
  1:  11  10
```

So we have available two sockets with 10 cores each (20 in total).

1 and 2 - Single Core

First of all I estimated the bandwidth of a single core of the node using the serial version of the code, and I ran it 10 times to get a mean time from this sample. To run it I used the command:

```
(for i in ‘seq 1 10‘; do OMP_NUM_THREADS=1 numactl --cpunodebind 0 --membind 0 ./s
```

were I used numactl to lock the process to socket 0 and local memory 0. The average result I obtained from the 10 runs is: 13486.70 ± 9.22 MB/s.
To compare the bandwidth reading from memory associated to the socket of core 1 against the memory associated with the other socket, I had to change the variable membind to 1, and the average of the 10 runs in this case is: 9824.84 ± 38.17 MB/s.

3 - Node

Then I had to estimate the overall bandwidth of one node of ulysess. In the case of my node, I had to use 20 threads (cause we have 20 cores in total). So I used the omp version of the code, but this time fixing the number of threads:

```
(for i in ‘seq 1 10‘; do OMP_NUM_THREADS=20 numactl --cpunodebind 0
--membind 0 ./stream_omp.x; done) | grep "Triad:" | awk ’{print␣$2}’
```

I ran the code 10 times and the average result I obtained is: 40252.38 ± 277.71 MB/s.

num threads	NUMA region 0 [Mb/s]	NUMA region 1 [Mb/s]
1	13423.6	9762.3
2	20151.3	17245.4
3	22012.0	20235.7
4	22051.2	21564.8
5	22156.3	21739.4
6	22426.2	22001.9
7	22019.6	21992.4
8	22361.2	22006.5
9	22112.0	22298.9
10	22094.7	22026.0

Table 1: Bandwidths I obtained changing the number of threads from 1 up to 10.

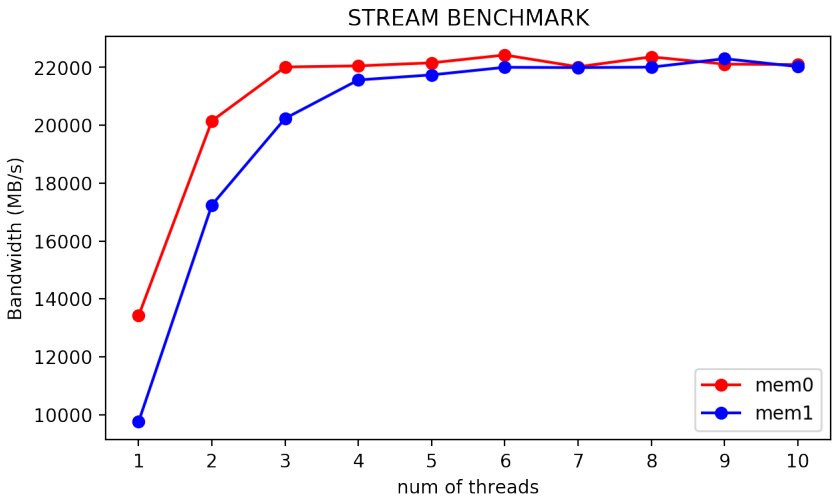

Figure 1: Bandwidths of the node c04-01

4 - All threads in one socket

Then I had to estimate the bandwidth of one socket of Ulysses. To do this I used the openmpi version of the code, changing this time the number of threads used (from 1 to 10) and I distinguished again the 2 cases of memory association: the one in which the memory was binded to the same socket (membind –0) and the one in which the process was forced to use distant memory (membind –1). To do this I used the following command line:

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

You can find the results of those runs in the table 1.
In figure 1 you can see the two curves of the bandwidths obtained varying the "membind" variable.

From the table and the figure we can clearly see that when a core has to use the memory associated to its socket it has an higher bandwidth if compared to the case in which it has to use the memory associated with the other socket.