Homework #8 NUMA Architecture and Hybrid OpenMP-MPI Programming
Due: Friday, April 20th, before 2:00pm

NUMA Architecture

You will be investigating how different NUMA settings impact memory access on JANUS by running different instances of the stream benchmark. Stream is a synthetic benchmark program that measures sustainable memory bandwidth (in MB/s). It performs the following:

1. `Copy:  c[k] = a[k] (one read and one write)`
2. `Scale: b[k] = alpha*c[k] (one read and one write)`
3. `Add:   c[k] = a[k] + b[k] (two reads and one write)`
4. `Triad: a[k] = b[k] + alpha*c[k] (two reads and one write)`

Download the `stream_numa.c` and the `stream_kmp.c` code from *https://learn.colorado.edu*. To compile the code, type:

use ICS

```
icc -openmp stream_numa.c –o stream_numa
icc -openmp stream_kmp.c –o stream_kmp_affinity
```

Questions

1. **stream_numa:** When you run stream on a single node using all 12 threads, you should see output similar to the following:

   ```
   export OMP_NUM_THREADS=12
   ./stream_numa
   ```

   ```
   Function      Rate (MB/s)   Avg time     Min time     Max time
   Copy:       13548.9979       0.0237       0.0236       0.0238
   Scale:      13285.2009       0.0241       0.0241       0.0242
   Add:        14807.5633       0.0325       0.0324       0.0325
   Triad:      14819.8803       0.0325       0.0324       0.0326
   ```

   This could be better. Try running the following numactl command:

   ```
   export OMP_NUM_THREADS=12
   numactl --interleave=all ./stream_numa
   ```

   How do you explain this improvement?

2. **stream_kmp**: Run the `stream_kmp` benchmark with 6 threads using two different KMP_AFFINITY settings:

```
export OMP_NUM_THREADS=6
export KMP_AFFINITY=compact
./stream_kmp
export KMP_AFFINITY=scatter
./stream_kmp
```

    a. Which setting works best?
    b. Why do you think this is true? HINT: consider running with KMP_AFFINITY=verbose,scatter (or compact) to see how threads are mapped to cores.

Hybrid OpenMP-MPI

Download the `trap.cpp` file from the *https://learn.colorado.edu*. *This is purely MPI code. To compile, type the following:*

use .openmpi-1.4.3_gcc-4.5.2_torque-2.5.8_ib

```
mpicxx -O2 -fopenmp trap.cpp -o trap
```

When run in with MPI, this code scales in the following way.

```
1    15.676
6    2.76263   ~x 5.6
12   1.30613   ~x 12.0
24   0.891787  ~x 17.6
36   0.44451   ~x 35.3
```

Add OpenMP support to this code and run it in the following ways:

1. OpenMP only with 6 threads
2. OpenMP only with 12 threads
3. Hybrid 4 MPI processes with 6 threads
4. Hybrid 6 MPI processes with 6 threads

Show your results and the commands you used to execute the code and compute the speedup for each instance. How do these compare with the MPI only results?

Assignment Submission

Create a PDF of your results and answers to your questions. Please work alone on this assignment and include your code from the hybrid OpenMP-MPI section at the end of your report. Submit this file to the d2L site by the due date.