Hands-on “MPI with Intel Xeon and Intel Xeon Phi Architecture”

1) Compile and run a MPI application on Xeon

```
mpiicc mpitest.c -o ~/mpitest mpirun -host localhost -n 10 ~/mpitest
```

1.1) Change the amount of ranks to 15 and execute again

```
mpirun -host localhost -n 15 ~/mpitest
```

2) Execute the application on two nodes (Xeon):

```
scp ~/mpitest phi03:~/ mpirun -host localhost -n 10 ~/mpitest : -host phi03 -n 10 ~/mpitest
```

3) Compile and run the code on Intel Xeon Phi.

Use variable I_MPI_MIC to enable the mic to execute MPI

```
export I_MPI_MIC=1
```

```
mpiicc mpitest.c -o ~/mpitest.mic -mmic mpirun -host mic0 -n 10 ~/mpitest
```

4) Execute the application on several Xeon Phi Coprocessors:

```
mpirun -host mic0 -n 10 ~/mpitest : -host mic1 -n 10 ~/mpitest : -host mic2 -n 10 ~/mpitest : -host mic3 -n 10 ~/mpitest
```

5) Execute the application on two Xeon nodes and several Xeon Phi Coprocessors:

```
mpirun -host localhost -n 10 ~/mpitest : -host phi03 -n 10 ~/mpitest : -host mic0 n 10 ~/mpitest : -host mic1 -n 10 ~/mpitest : -host mic2 -n 10 ~/mpitest : -host mic3 -n 10 ~/mpitest
```

6) MPI Pinning

```
export I_MPI_DEBUG=4 mpirun -host localhost -n 10 ~/mpitest
```

Note that after executing the application a pinning / report is generated on the standard output:

```
[0] MPI startup(): Rank    Pid     Node name          Pin cpu
[0] MPI startup(): 0      50344   phi02.ncc.unesp.br  {0,1,2,3,36,37,38}
[0] MPI startup(): 1      50345   phi02.ncc.unesp.br  {4,5,6,39,40,41,42}
[0] MPI startup(): 2      50346   phi02.ncc.unesp.br  {7,8,9,10,43,44,45}
[0] MPI startup(): 3      50347   phi02.ncc.unesp.br  {11,12,13,46,47,48,49}
[0] MPI startup(): 4      50348   phi02.ncc.unesp.br  {14,15,16,17,50,51,52}
[0] MPI startup(): 5      50349   phi02.ncc.unesp.br  {18,19,20,53,54,55,56}
[0] MPI startup(): 6      50350   phi02.ncc.unesp.br  {21,22,23,24,57,58,59}
[0] MPI startup(): 7      50351   phi02.ncc.unesp.br  {25,26,27,60,61,62,63}
[0] MPI startup(): 8      50352   phi02.ncc.unesp.br  {28,29,30,31,64,65,66}
[0] MPI startup(): 9      50353   phi02.ncc.unesp.br  {32,33,34,67,68,69,70}
```

6.1) group resources by domain

```
export I_MPI_PIN_DOMAIN=node mpirun -host phi02 -n 10 ~/mpitest
```

6.1) group resources by socket

```
export I_MPI_PIN_DOMAIN=socket mpirun -host phi02 -n 10 ~/mpitest
```

7) MPI and OpenMP Pinning

```
mpiicc mpiOpenMPtest.c -o ~/mpiOpenMPtest -fopenmp mpiicc mpiOpenMPtest.c -o
~/mpiOpenMPtest -fopenmp -mmic
```

```
export OMP_NUM_THREADS=2 export KMP_AFFINITY=verbose,scatter export
I_MPI_PIN_DOMAIN=socket
```

```
mpirun -host phi02 -n 2 ~/mpiOpenMPtest
```

Note that each rank will fork two threads using only the resources of its domain

Define different mpi/openmp affinity in each rank

```
mpirun -env KMP_AFFINITY=verbose,scatter -env OMP_NUM_THREADS=2 env
I_MPI_PIN_DOMAIN=cache1 -host phi02 -n 2 ~/mpiOpenMPtest : -env OMP_NUM_THREADS=4 -env
I_MPI_PIN_DOMAIN=cache1 -env KMP_AFFINITY=verbose,scatter -env
LD_LIBRARY_PATH=/opt/intel/lib/mic/ -host mic0 -n 2 ~/mpiOpenMPtest
```

8) Profilling a MPI Application using itac

8.1) compile prime number application

```
mpiicc prime_mpi.c -o prime_mpi mpiicc prime_mpi.c -o prime_mpi.mic -mmic
```

8.2) create directories for traces files:

```
mkdir ~/traces mkdir ~/traces/host mkdir ~/traces/mic mkdir ~/traces/mics
```

8.3) put -trace flag on mpirun

8.3.1) Execute prime_mpi on host

```
cd ~/traces/host mpirun -trace -host localhost -n 10 ~/prime_mpi
```

8.3.2) Execute prime_mpi on mic

```
cd ~/traces/mic mpirun -trace -host mic0 -n 10 ~/prime_mpi.mic
```

8.3.3) Execute prime_mpi on two mics

```
cd ~/traces/mics mpirun -trace -host mic1 -n 5 ~/prime_mpi.mic : -host mic1 -n 5
~/prime_mpi.mic
```

8.4) open MPI Trace Analyzer application

traceanalyzer click on file to open a result open the stf files on the following directories:

```
~/traces/host ~/traces/mic ~/traces/mics
```

open the result file:

How many time was spent with communication and with computation in each test?

What was the costly MPI function in each test?