Hands-on

Execute the following steps before start exercices 1 and 2:

ssh -X 172.16.64.4

cd ~/

git clone https://github.com/intel-unesp-mcp/multiprogramming-Intel.git

source /opt/intel/parallel_studio_xe_2018/psxevars.sh intel64

1. Identifing the computational resources of KNL:

Execute the comand **lscpu**

what are the amount of processors / cores?

How much memory is available at each cache level?

Execute the comand **numactl -H**

How many nodes are available?

2. Executing multiplication transposition[1] on KNL using different memory systems:

Execute the following commands:

cd ~/multiprogramming-Intel/hands-on/knl/3

make clean

make runme-CPU

Execute the matrix transposition using mcdram

time numactl -m 0 ./runme-CPU 15000 100

Execute the matrix transposition using dram

time numactl -m 1 ./runme-CPU 15000 100

For this matrix transposition which memory system has better performance?

[1] https://colfaxresearch.com/multithreaded-transposition-of-square-matrices-with-common-code-for-intel-xeon-processors-and-intel-xeon-phi-coprocessors/3. Controlling the MKL execution

Execute the following steps before start exercices 3, 4 and 5:

ssh -X 10.20.230.113

cd ~/

git clone https://github.com/intel-unesp-mcp/multiprogramming-Intel.git

source /opt/intel/parallel_studio_xe_2018/psxevars.sh intel64

3. Identifing the computational resources of Xeon:

Execute the comand **lscpu**

what are the amount of processors / cores?

How much memory is available at each cache level?

4) OpenMP: Compiling, running and environment variable

The file hello_omp.c implements an application that uses OpenMP pragmas:

4.1 Compile hello_omp.c to Intel Xeon:

```
icc hello_omp.c -o hello_omp -fopenmp
./hello_omp
```

4.2 Execute the code on Intel Xeon with 16 threads

```
export OMP_NUM_THREADS=16
./hello_omp
```

5) Thread affinity

5.1 Execute hello_omp with 10 threads and using affinity policy to allocate threads close to each other (compact)

```
export KMP_AFFINITY=compact,verbose
export OMP_NUM_THREADS=10
./hello_omp
```

5.2 Execute hello_omp with 10 threads and using affinity policy to spread threads among processors (scatter)