Hands-on “Optimizing Machine Learning Applications using Intel Architectures”

Execute the following steps before start exercices 1 and 2:

```
ssh -X phi05.ncc.unesp.br
cd ~/
git clone https://github.com/intel-unesp-mcp/workshop-HPC-ML.git
source /opt/intel/parallel_studio_xe_2017.1.043/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 ~/workshop-HPC-ML/knl/3
make clean
make runme-CPU
```

Execute the matrix transposition using mcdram

```
time numactl -m 4,5,6,7 ./runme-CPU 15000 100
```

Execute the matrix transposition using dram

```
time numactl -m 0,1,2,3 ./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 phi02.ncc.unesp.br
cd ~/
git clone https://github.com/intel-unesp-mcp/workshop-HPC-ML.git
source /opt/intel/parallel_studio_xe_2017.1.043/psxevars.sh intel64
```

1) Compiling running and environment variable

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

1.1 Compile hello_omp.c to Intel Xeon:

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

1.3 Execute the code on Intel Xeon with 16 threads

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

2) Thread affinity

2.1 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
```

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