*Hands-on “OpenMP with Intel Xeon and Intel Xeon Phi Architecture”*

### 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
```

**1.2 Compile hello_omp.c to Intel Xeon Phi**

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

**1.3 Execute the code on Intel Xeon with 16 threads**

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

**1.4 Execute the code on Intel Xeon Phi with 16 threads**

```
ssh mic0
export  OMP_NUM_THREADS=16
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/intel/lib/mic/
./hello_omp.mic
```

### 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)**

**2.3 Execute hello_omp with 10 threads and using affinity policy to spread threads among processors on Xeon Phi (scatter)**

**2.4 Execute hello_omp with 10 threads and using affinity policy to balance the thread allocation among processors on Xeon Phi (balanced)**

**3) vectorization using OpenMP #pragma omp declare simd**

3.1 Compile the file OMP4-7.c with compilation report:

```
icc OMP4-7.c -o OMP4-7 -fopenmp -vec-report6

cp OMP4-7.optrpt OMP4-7.optrpt2
```

3.2 Include #**pragma omp declare simd** in top of functions **min** and **distsq** and compile OMP4-7.c again:

```
icc OMP4-7.c -o OMP4-7 -fopenmp -vec-report6
```

3.3 Compare the compilation report of both compilations, and verify whether the functions could be vectorized:

```
diff OMP4-7.optrpt OMP4-7.optrpt2
```

**Task 4 data transfer between processor and coprocessor**

In the code **omp4.c** a set of functions and variable have been declared to be available in Xeon and Xeon phi Memory System using OpenMP pragmas from line 4 – 18.

4.1 ) Implement in this code the following sequence of commands (after line 27):

- Copy variables a, b and c to device 3 using omp update
- Execute sum2 with value a, b and c on device 3
- Copy variable sum from device 3 using omp update
- Print the value of variable sum on host
- Copy variable sum to device 2 using omp update
- Execute multiply2 with value sum2 on device 2
- Copy variable mult from coprocessor using omp update
- Print the value of variable mult on host

4.2 )Turn offload report on:

```
export OFFLOAD_REPORT=2
```

4.3) Compile and Execute the application