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

1. Intel Xeon Phi Management

1.1. Execute the following command to verify if the service that controls the devices is up and running:

```
service mpss status
```

1.2. Execute the following command to obtain information about the devices:

```
micinfo or mpssinfo
```

How Many Intel Xeon Phi devices are deployed?

1.3. Execute the following commands in the main host and on one mic device.

- The following command returns the number of cores
  - `cat /proc/cpuinfo | grep 'cpu cores' | uniq`
- The following command returns the number of threads
  - `cat /proc/cpuinfo | grep processor | wc -l`

How many cores and threads is available on main host and on mic Device?

2. Intel Xeon and Intel Xeon Phi Compiling and Running

The code **helloWord.c** shows the amount of logical threads available.

2.1. Compile and run in Intel Xeon using the following commands:

```
icc helloWord.c -o helloWord
./helloWord
```

2.2. Compile to Intel Xeon Phi:

```
icc helloWord.c -o helloWord.mic -mmic
```

2.3. Run in Intel Xeon Phi using micnativeloadex:

```
micnativeloadex helloWord.mic
```

2.4. Run in Intel Xeon Phi using SSH:

```
cp helloWord.mic ~/
ssh mic0
~/helloWord.mic
```

3. Offload

The code **helloWordOffload.c** performs the offload of a region of code to Intel Xeon Phi.

3.1. Compile and run on Intel Xeon using the following commands:

```
icc helloWordOffload.c -o helloWordOffload
./helloWordOffload
```

3.2. Debug the offload using variable OFFLOAD_REPORT and run again:

```
export OFFLOAD_REPORT=2
./helloWordOffload
```

3.3. Change the code **helloWordOffload.c**. Add another region of code to be executed on Intel Xeon Phi, that performs the sum of two double elements and display the result.

Use the following snippet:

```
double A, B, sum;
A=3;
B=5;
sum  = A + B;
printf("Result : %f", sum);
```

3.4. Change the device to offload code using the following parameter **(mic:deviceId)**

```
#pragma offload target(mic:2)
```

4. Offload and Data Transfer

The code **offloadFunction.c** shows a function that transfers function and data to be executed on Intel Xeon Phi:

4.1. Compile and run on Intel Xeon using the following commands:

```
icc offloadFunction.c -o offloadFunction
./offloadFunction
```

How much data was transferred from CPU to MIC and from MIC to CPU?

4.2. Change the code **offloadFunction.c** Add a function called MyFunction2 to be executed on Intel Xeon Phi, that performs the sum of all elements of an array of double elements, and display the value of sum on the host. (transfer variable C[] from host to device and variable sum from device to host)

Use the following snippet:

```
  int cont;
  int n=100;
  for (cont=0; cont<n; cont++)
    sum += C[cont];
```