Hands-on "OpenMP with Intel Xeon and Intel Xeon Phi Architecture"

1 Compiling running and environment variable

The file hello_omp.c implements an example of openMP4:

1.1 Compile hello_omp.c to Intel Xeon

1.2 Compile hello_omp.c to Intel Xeon Phi

1.3 Execute the code on Intel Xeon with 16 threads

1.4 Execute the code on Intel Xeon Phi with 100 threads

2 Thread affinity

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

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 Use pragma omp target to execute method SUM two times. First on device 3 using parameterx=3 and then on device 2 using parameter x=2. Using the code below:

```
#include <stdio.h>
#include <unistd.h>

void sum(int x) {
  printf("host has %ld logical cores.\n", sysconf(_SC_NPROCESSORS_ONLN ));

  #pragma omp target if (i==3) device(2) map(to:x)
  {
    printf("mic has %ld logical cores. i = %d\n", sysconf(_SC_NPROCESSORS_ONLN ), i);
  }
}
int main() {

//put your code here…

}
```

Task 4 : pragma update

Include in the code below the following sequence of commands:

- Copy variables A, B and C to CoProcessor using omp update
- Execute sum2 with value A, B and C on device 3
- Copy variable sum from CoProcessor using omp update
- Execute multiply2 with value sum2 on device 2
- Copy variable mult from CoProcessor using omp update

```
#include <stdio.h>
#include <unistd.h>

#pragma omp declare target

int sum,mult;

int sum2(int a, int b, int c) {
     sum=a+b+c;
}

int multiply2(int res) {
     mult=sum*sum;
}

#pragma omp end declare target

int main() {
  int a,b,c;
  a=rand();
  b=rand();
  c=rand();
  sum=0;
  mult=0;

  put your code here...
}
```

Task 5 # pragma omp declare simd

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
```

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
```

Compare the compilation report of both compilations, and verify weather the functions could be vectorized.

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