HPML Lab 1

Theoretical questions are identified by Q<number> while coding exercises are identified by C<number>. Submit a tar-archive named with your nyu-net-id (e.g. <net-id>.tar => uf5.tar) that unpacks to
<net-id>/dp1.c
<net-id>/dp2.c
<net-id>/dp3.c
<net-id>/dp4.py
<net-id>/dp5.py
<net-id>/results.pdf

The *<net-id>/results.pdf* contains the outputs of the programs and the answers to the questions.

C1 (6 points):

Write a micro-benchmark that investigates the performance of computing the dot-product in C that takes two arrays of 'float' (32-bit) as input. The dimension of the vector space and the number of repetitions for the measurement are command-line arguments, i.e. a call 'dp1 1000 10' performs 10 measurements on a dot product with vectors of size 1000. Initialize fields in the input vectors to 1.0.

```
float dp(long N, float *pA, float *pB) {
  float R = 0.0;
  int j;
  for (j=0;j<N;j++)
    R += pA[j]*pB[j];
  return R;
}
```

Name the program dp1.c and compile it with
gcc -O3 -Wall -o dp1 dp1.c

Make sure the code is executed on a Greene node with Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz or similar and enough RAM, the 300000000 size runs should not be killed by the system!

Measure the execution time of the function with clock_gettime(CLOCK_MONOTONIC).
Measure the time for N=1000000 and N=300000000.
Perform 1000 repetitions for N=1000000 and 20 repetitions for N=300000000.
Compute the appropriate mean for the execution time for the second half of the repetitions.

For the average times, compute the bandwidth in GB/sec and throughput in GFLOP/sec, print the result as

N: 1000000 <T>: 9.999999 sec B: 9.999 GB/sec F: 9.999 GFLOP/sec
N: 300000000 <T>: 9.999999 sec B: 9.999 GB/sec F: 9.999 GFLOP/sec

where,
N - Number of repetitions
<T> - mean for the execution time for the second half of the repetitions
B - bandwidth computed using the mean for the execution time for the second half of the repetitions
F - FLOPS computed using the mean for the execution time for the second half of the repetitions

Please attach the first few lines of your code output to *<net-id>/results.pdf.*

C2 (3 points):

Perform the same microbenchmark with

```
float dpunroll(long N, float *pA, float *pB) {
  float R = 0.0;
  int j;
  for (j=0;j<N;j+=4)
    R += pA[j]*pB[j] + pA[j+1]*pB[j+1] + pA[j+2]*pB[j+2] + pA[j+3] * pB[j+3];
  return R;
}
```

C3 (3 points):

Perform the same microbenchmark with MKL (Intels library), you may need to install a 'module' on the greene to access MKL.

```
#include <mkl_cblas.h>
float bdp(long N, float *pA, float *pB) {
  float R = cblas_sdot(N, pA, 1, pB, 1);
  return R;
}
```

C4 (6 points):

Implement the same microbenchmark in python, using NumPy arrays as input

```
A = np.ones(N,dtype=np.float32)
B = np.ones(N,dtype=np.float32)

for a simple loop
def dp(N,A,B):
   R = 0.0;
   for j in range(0,N):
      R += A[j]*B[j]
   return R
```

C5 (4 points):

Perform the above measurements (C4) using 'numpy.dot'.

Q1 (3 points):

Explain the consequence of only using the second half of the measurements for the computation of the mean.

Q2 (6 points):

Draw a roofline model based on 200 GFLOPS and 30 GB/s. Add a vertical line for the arithmetic intensity.
Draw points for the 10 measurements for the average results for the microbenchmarks. (For each benchmark you have to plot two cases, one for 100000 and the other for 300000000. So, in total for 5 benchmarks there will be 10 measurements.)

Q3 (5 points):

Using the N=300000000 simple loop as the baseline (C1), explain the
the difference in performance for the other 2 measurements in the C variants (C2 and C3).

Q3 (6 points):

Check the result of the dot product computations (C5) against the analytically calculated result (C1).
Explain your findings. (Hint: Floating point operations are not exact.)

NOTE: As a tip for the double-log chart for the roofline model, you can use a tool like gnuplot. Note that a bandwidth line may not reach an X axis plotted at Y=0!