High Performance Computing

Assignment 2

The goal of this assignment is to use multiple threads for performing matrix multiplication. We will do this by adding OpenMP directives to your codes from Assignment 1, and by using multithreaded BLAS libraries.

(a) **10 marks.** Add OpenMP directives to parallelize your three versions `matmult` developed in Assignment 1. You may make corrections to your Assignment 1 codes if necessary. After adding the OpenMP directives, check your codes for correctness as you did in Assignment 1.

(b) **10 marks.** For two different problem sizes (try $n = 500$ and n as large as possible), measure timings using different numbers of threads. On Intel Xeon Phi, use 1 to 240 threads.

Plot the timings (as a function of number of threads) for the three versions of `matmult` on the same graph. In a second graph, plot the speedup of the three versions. In a third graph, plot the parallel efficiency of the three versions. Finally, in a fourth graph, plot the computation rate (in Gflop/s). All graphs have the x-axis being the number of threads. There should be 8 graphs, four for $n = 500$ and four for your other value of n.

(c) **10 marks.** Instead of using your own matrix multiplication code, we will now test the performance of different implementations of the `dgemm` function. It will be interesting to see how the performance of your implementations compare to the performance of the optimized functions!

A test program called dgemm-bench is available on the web. You will need to link this program to the BLAS library. You can also try linking to the Intel Math Kernel Library (MKL). For the latter, you can use the example makefile provided to you in the mkl directory of our course repository.

For the three versions of dgemm on two different architectures, plot the computation rate (in Gflop/s) as a function of the number of threads (for 1 to 36 threads) on the same graph. Use n at least 3000.

Finally, for one of the three versions of dgemm, plot the computation rate (in Gflops/s) as a function of n, the matrix size.

(d) **5 marks.** Submit a short report of your results as a pdf file on T-square. Marks will be given for the clarity of your report, labeling your graph axes, etc. Briefly explain why you think your results are reasonable. Include a listing of your code for part (a) in the pdf file. How does the performance of your code compare to the performance of the dgemm functions? What is the influence of the problem size n? How does the performance of dgemm (both OpenBLAS and MKL versions) compare to the peak floating point performance on Intel Xeon Phi?