CS294 - Homework Assignment 5
Performance Debugging
Due date: 11:59 PM, November 13, 2017

November 2, 2017

In this assignment you will take an existing implementation of multigrid, and perform the steps required to assess and improve its performance. In the directory `resources/homework5` you will find a makefile and and standard directory setup. In particlar, you will find in `src/mg` the source `MultigridClass.`{`H,cpp`}, and in `/exec`, `multigridTest.cpp`. If you set your shell environment variable `CH_TIMER`, the timers already in the program will be invoked, and a file `time.table` will be generated. In addition, a estimate of the number of flops computed in multigrid is output to `cout`.

1. Build the test program (the make target `testMultigrid`) and run it with

   ```
   input log_2(domainSize)
   8
   input number of multigrid levels
   8
   input max number of iterations, convergence tolerance
   15 1.e-10
   ```

   Looking at the timings for `mg` in `time.table` and the flop counts produced by the program, estimate the flop rate the program is running at. Compare that to the timings produced in homework 3 for your naive implementation of `dgemm`. How much would you reasonably expect to improve the performance by ?

2. Reimplement one or more member functions in `Multigrid` to improve data locality using the pencil construction discussed in class. The output for the program should be identical to that of the original program. Some hints:

   - You know the data layout is a `RectMDArray` is row-major (incrementing by one in the first index is stride one).
   - `double* rmda_ptr = &RMDA[RMDA.getDBox().getLowCorner()]` gives you access to the contiguous block of data in which the valuesof `RectMDArray<double> RMDA` are stored.

- In the program there are `#if...#else...#endif` preprocessor commands in each of the main member functions of `Multigrid`. Right now, they are given as `#if 0`, which means that the first block after the `#if` is skipped, and the baseline implementation is compiled. If you change `#if 0` to `#if USE_PENCIL`, it will allow you to use the first block or not, depending on whether you set `USE_PENCIL = 1` or `0` in your makefile. Inserting your optimized code with the `USE_PENCIL` flag will allow you to more easily compare the optimized vs. non-optimized code.

3. You will submit in the `doc/` directory a document describing what you did, and comparing the performance of the baseline with that of your improved code, as well as submitting the final version of your implementation. We will also run the improved version that you submit. As part of your report, you should tell us the properties of the system you ran on. On a Linux machine, this the output from the command `lscpu`. On a Mac, you can find this in
   `<Apple> -> about this Mac-> System Report`,
   which takes you to the hardware overview. You should also tell us the what compiler you used to obtain your results, as well as the version number. This can be found by invoking the compiler e.g. `clang++ --version`, `g++ --version`.