

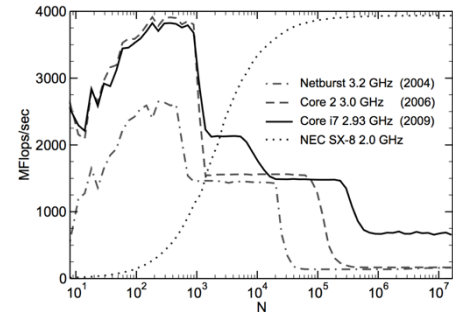
CMSE/CSE 822: Parallel Computing
Fall 2019, Homework 3
Due 11:59 pm, Monday, October 7th

Important note: Please use a word processing software (e.g., MS Word, Mac Pages, Latex, etc.) to type your homework. Follow the submission instructions at the end to turn in an electronic copy of your work.

1) [50 pts] Compilers and auto-vectorization

As discussed over the last few lectures, writing high performance software in a high-level programming language partially depends on the compiler's ability to generate optimized machine code. In this problem, you are asked to analyze this aspect of high performance computing in the context of the vector triad kernel.

- a. For the system you are running on (i.e., an intel16 node), determine the clock speed of the processor and the cache size/layout by looking up its specifications (HPCC Wiki and Intel's website are useful references, or alternatively you may use Unix commands such as `lscpu`). Assuming the processor is capable of one floating point operation per clock cycle, what would you expect the theoretical peak performance of this system to be?
- b. Now consider the fact that modern CPUs support wide vector instructions (SSE, AVX, etc.) and the fused multiply add operation (FMA: https://en.wikipedia.org/wiki/FMA_instruction_set). How does this change your expectations regarding the peak CPU performance of part (a)?
- c. Using the provided `vector_triad.c` file, plot the performance of the vector triad kernel at several vector lengths N (similar to the one shown on the right but in Gflop/s) using different compilers and options (see below). On these plots, place horizontal lines representing your theoretical peak performance estimations from parts (a) & (b) (i.e., 1. assuming one flop/cycle, 2. with SSE vectorization, 3. with AVX vectorization, and 4. using FMA).



You are expected to generate two plots, one using the GCC compiler, another using the Intel compiler both of which are available at HPCC. For GCC, experiment with compiler options such as `-O1`, `-O3`, `-O3 -march=native`. For Intel, experiment with options such as `-O1`, `-O3`, `-O3 -xSSE`, `-O3 -xAVX`, `-fast`.

- d. How does the maximum measured performance using different compilers & options compare to the different peaks you estimated? Can you tell which compiler options have enabled what kind of optimizations?
- e. Are there any sudden features in your plot? Explain them in the context of the hardware architecture (i.e., cache layout) of your system.

2) [50 pts] Stream benchmark and memory bandwidth

Besides the peak CPU performance, another important factor impacting performance is the memory bandwidth as we discussed within the context of the Roofline model. In this problem, you are asked to analyze the memory performance of your system.

- a. Download, compile, and run the STREAM memory bandwidth benchmark (<http://www.cs.virginia.edu/stream/>). Report the bandwidth computed by STREAM for your system (i.e., Intel16 at HPCC). *Note: For properly using the software, it is highly recommended to read the instructions and comments included within the source code.*
- b. Write a simple program (in C, C++, or Fortran) for performing a vector copy operation, e.g. in C:

```
double c[n], a[n];
for (k=0; k<ntrials; k++) {
    for (i=0; i<n; i++)
        c[i] = a[i];
}
```

Note that you can use the `vector_triad.c` code from Problem 1 to start off, and modify it as necessary. Measure the memory bandwidth of your code, in GB/s, for $n=2000000$ and compare this to the STREAM result. What factors may impact the performance of your code?

- c. Now run your code for various values of n (for instance $n=1, 16, 1024, 16384, 131072, 1048576$, etc.), again using different compilers and options as in Problem 1. Plot the results for memory bandwidth versus n . Also modify the STREAM benchmark to run at the same values of n and include this also on your plot. How does your simple vector copy code compare to STREAM results?
- d. Discuss how the memory bandwidth performance varies with n and why. In particular, do you observe regions where your measured memory bandwidth is much higher than what it should be? Can you use the information in such regions to estimate the bandwidth from various levels of the cache?

Instructions:

- **Compiling your programs:** No makefiles are provided in this assignment. You are expected to do manual compilation or write up your own makefiles (you can use the one in the previous assignment for starter). A simple way to do manual compilation with gcc is as follows:

```
gcc -O3 vector_triad.c -lm
```

Note: This command will compile the vector_triad.c file with -O3 level of optimization; *-lm* option is needed to link with the math library. The resulting executable will be named “a.out” which can be modified by using the “-o” compiler flag.

- **Measuring your execution time and performance properly:** The wall-clock time measurement mechanism (based on the gettimeofday() function) implemented in the provided source file will allow you to measure the timings for a particular part of your program (see the skeleton code) precisely. However, while they are convenient to use, the dev-nodes will have several other programs running simultaneously, and your measurements may not be very accurate due to the “noise”. After making sure that your program is bug-free and executes correctly on the dev-nodes, a good way of getting reliable performance data for various input sizes is to use either the interactive queue or the batch jobs queue. Please refer to instructions in HW2 for guidance on how to use HPCC in these two different modes of execution. **Also, please use the intel16 cluster for all your performance measurement runs!**
- **Submission:** Your submission will include only one file:
 - A pdf file named “HW3_yourMSUNetID.pdf”, which contains your answers to the questions in the assignment.

Further instructions regarding the use of the git system and homework submission is given in the “Homework Instructions” under the “Reference Material” section on D2L. Make sure to strictly follow these instructions; otherwise you may not receive proper credit.