Designing and building applications for extreme scale systems
CS598 SP2016
HW6: Stream with Threads

Goals

- Gain experience with OpenMP
- Learn about memory performance with threads and compare to simple performance models

Tasks

Part 1

Modify your program from HW2 to use multiple threads for the copy operation. That is, write a program that measures the performance of this operation (in C):

```
double c[n], a[n];
for (k=0; k<nk; k++) {
        // Run the following loop with Nt threads
        for (i=0; i<n; i++)
            c[i] = a[i];

        //See note below
        }
```

This is a simple copy from an array a to an array c. Fill in the following table for two measurements:

- Time for loop (s)
- Rate (MB/s)

Nt	Time for loop (s)	Rate (MB/s)
1		
2		
4		
6		
8		
10		
12		
14		
16		

Note 1

Depending on your compiler and the optimization settings, you may need to keep the compiler from eliminating code that it believes unnecessary. Add a call to a routine

```
void dummy(double *a, double *c) {}
```

After the copy loop but within the loop over k. Place this routine in a separate file (that keeps the compiler from finding and eliminating it as well). Use a value of n that is large enough that the arrays a and c do not fit into cache.

When running your program, make a separate run of the program for each number of threads (Nt in the table below). You should be able to use

```
export OMP_NUM_THREADS=10
```

to set the number of threads to 10 (assuming the sh or bash shell).

Note 2
A MB is a Megabyte = 10^6 bytes (note: 2^{20} bytes is a Mebibyte, written MiB, not a Megabyte). To time the loop, you may use any accurate timer. You may also look at the code in Part 2 and use the same timer used for that code. You may use the timer provided by OpenMP, omp_get_wtime(), which returns a double value of seconds since a fixed (but unspecified) time in the past.

When timing the loop, pick nk such that the time measured for both loops is about one second. Report the total time divided by nk as the "Time for loop". When computing the rate, count each byte written or read as a separate byte. I.e., for n=10 and for 8 byte doubles, this copy loop moves n*8*2=10*8*2=160 bytes.

In some cases, the OpenMP runtime may create new threads at the beginning of each OpenMP section, which can be time consuming.

Part 2
Compare your results to the performance mode of rate = min(Nt * r_1, r_{max}), where Nt is the number of threads, r_1 is the rate for a single thread (using OpenMP), and r_{max} is determined by inspection. Comment on the fit. For how many threads does Nt * r_1 = r_{max}?

Submit
For **Part 1**, plot the results, using a linear plot. Turn in both the table and the plot. Also report on which system you ran your test. Submit your code with your table. For **Part 2**, hand in necessary discussion and comments.

Things to think about (but not hand in)
1. How would you fit the data? How accurate are your measurements?
2. How would you test smaller array sizes (so that they fit in cache), taking into account the time cost of starting an OpenMP parallel for?
3. What does your result imply about the performance of a sparse matrix-vector multiply running on each core?