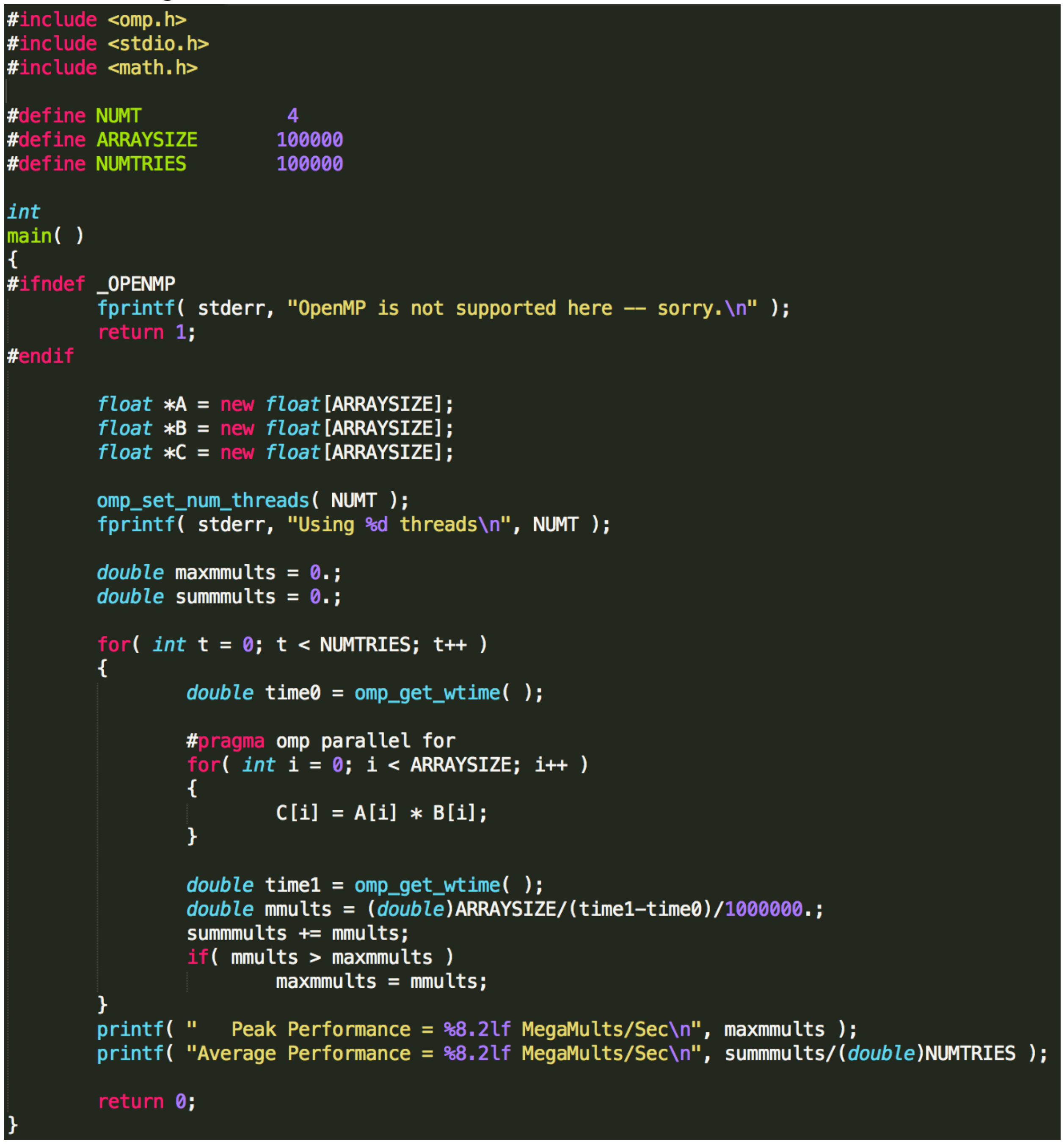
Parallel Programming

CS575

Chao Zhang

Project #0

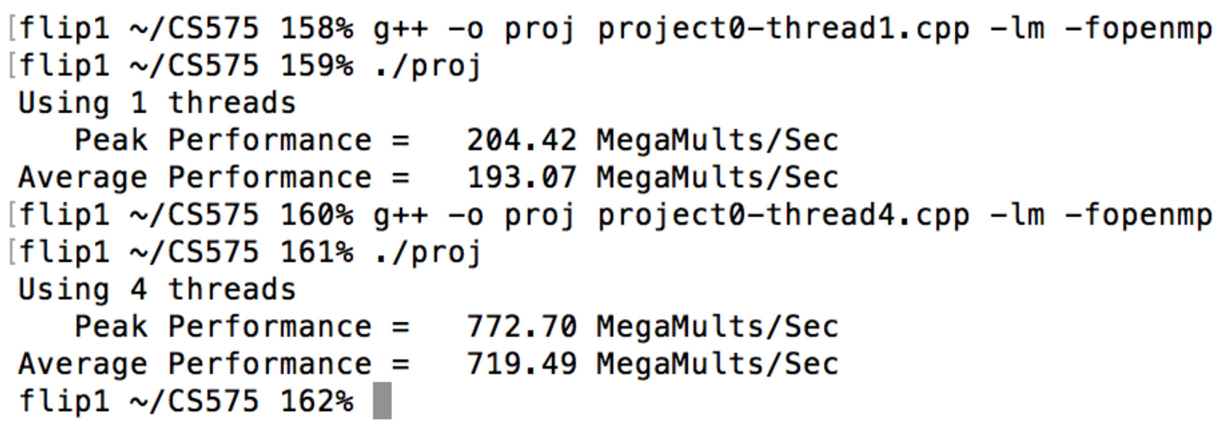
1. Source listings



```
#include <omp.h>
#include <stdio.h>
#include <math.h>

#define NUMT             4
#define ARRAYSIZE       100000
#define NUMTRIES        100000

int
main( )
{
#ifndef _OPENMP
        fprintf( stderr, "OpenMP is not supported here -- sorry.\n" );
        return 1;
#endif

        float *A = new float[ARRAYSIZE];
        float *B = new float[ARRAYSIZE];
        float *C = new float[ARRAYSIZE];

        omp_set_num_threads( NUMT );
        fprintf( stderr, "Using %d threads\n", NUMT );

        double maxmmults = 0.;
        double summmults = 0.;

        for( int t = 0; t < NUMTRIES; t++ )
        {
                double time0 = omp_get_wtime( );

                #pragma omp parallel for
                for( int i = 0; i < ARRAYSIZE; i++ )
                {
                        C[i] = A[i] * B[i];
                }

                double time1 = omp_get_wtime( );
                double mmults = (double)ARRAYSIZE/(time1-time0)/1000000.;
                summmults += mmults;
                if( mmults > maxmmults )
                        maxmmults = mmults;
        }
        printf( "   Peak Performance = %8.2lf MegaMults/Sec\n", maxmmults );
        printf( "Average Performance = %8.2lf MegaMults/Sec\n", summmults/(double)NUMTRIES );

        return 0;
}
```

I give the array size as 100000 and it will loop 100000 times. The function double time1 = omp_get_wtime() will gives me the wall clock time in second.

2. Results



```
[flip1 ~/CS575 158% g++ -o proj project0-thread1.cpp -lm -fopenmp
[flip1 ~/CS575 159% ./proj
 Using 1 threads
    Peak Performance =   204.42 MegaMults/Sec
 Average Performance =   193.07 MegaMults/Sec
[flip1 ~/CS575 160% g++ -o proj project0-thread4.cpp -lm -fopenmp
[flip1 ~/CS575 161% ./proj
 Using 4 threads
    Peak Performance =   772.70 MegaMults/Sec
 Average Performance =   719.49 MegaMults/Sec
 flip1 ~/CS575 162% 
```

This result I got was running on the flip server. It is the Linux system. From the result, I can see that the speed of 4 thread is almost 4 times than 1 thread. The 1 thread time is 204.42 MegaMults/Sec and the 4 thread time is 772.70. The reason of this is because the thread of 4 is 4 times than 1 thread, so the speed should be near the 4 times. For each result of the 1 and 4 thread, the peak performance is near the average performance. So the timing can be considered as reliable.