Project #0

Paige Lorson

April 5, 2019

Introduction

A great use for parallel programming is identical operations on large arrays of numbers. The goal of this project is to test the multiplication of two vectors with different numbers of threads and comment on the results.

Results

The machine used to produce the following results is a HP Z240 SFF Workstation. The calculation was made with an ARRAYSIZE of 1,000,000 and a NUMTRIES of 100. This produced the following peak performance results.

1 Thread	4 Threads
509.27 MegaMults/Sec	1927.64 MegaMults/Sec

From these results the speedup, defined as the ratio of Execution time with one thread to the Execution time with four threads, is calculated to be

$$S = 3.785$$

With the same amount of work divided amongst four workers, it may be expected to see a factor of 4x as the speedup. The difference between what was expected and the actual speedup may be attributed to the over head needed to setup four parallel threads.

The Parallel Fraction for this test is found so be

$$F_P = 0.981$$

Appendix

Listing 1: C++ code using listings

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

#define ARRAYSIZE          1000000 // you decide
#define NUMTRIES           100 // you decide
#define N                  1

int
main( )
{
#ifndef _OPENMP
        fprintf( stderr, "OpenMP is not supported here -- sorry.\n" );
        return 1;
#endif
// float N[] = {1, 4};
// double maxMegaMults[2] = {0., 0.};
// for( int j = 0; j < 2; j++ )
// {
        float A[ARRAYSIZE];
        float B[ARRAYSIZE];
        float C[ARRAYSIZE];

        // #define NUMT    N[j]
        omp_set_num_threads( N );
        fprintf( stderr, "Using %d threads\n", N );

        double maxMegaMults = 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 megaMults = (double)ARRAYSIZE/(time1-time0)/1000000.;
                if( megaMults > maxMegaMults )
                        maxMegaMults = megaMults;
        }

        printf( "Peak Performance = %8.2lf MegaMults/Sec\n", maxMegaMults );
// }
// double S = maxMegaMults[0]/maxMegaMults[1];
// printf( "Speedup = %8.3lf \n", S );
//
// double Fp = (4./3.)*( 1. - (1./S) );
// printf( "Parallel Fraction = %8.3lf\n", Fp );
```

```
53      // note: %lf stands for "long float", which is how printf prints a "double"
54      //        %d stands for "decimal integer", not "double"
55
56          return 0;
57  }
```