Project #2

OpenMP: Static vs Dynamic and Small vs. Large Chunksize

100 Points

Due: May 1

Introduction

This project involves an investigation of OpenMP scheduling. You will create a rumble between static scheduling vs. dynamic, and small vs. large chunksize.

You are deliberately giving each thread a very different amount of work to do and testing how OpenMP can best accomplish that.

Requirements

- Use OpenMP for this. Have a global floating-point Array whose size, ARRAYSIZE, is at least 32K (32*1024).
- Use a variety of different numbers of threads. Use at least 1, 2, and 4. You will get better results if you use more, maybe something like 1, 2, 4, 6, 8, 10, 12, 14, and 16. You can also use more if you'd like.
- Fill that array with ARRAYSIZE random floating-point numbers, e.g.,

```
unsigned int seed = 0;
Array[i] = Ranf( &seed, -1.f, 1.f );
```

Why do this if we don't care about the answers? This keeps a smart compiler from trying to figure out what the answers are ahead of time and avoiding the computation altogether. Maybe I'm just paranoid.

- The outer for-loop loops through i = 0 ... ARRAYSIZE-1 iterations. For each of these iterations, an inner for-loop loops through j = 0 ... i and that many of the Array's floating-point numbers are multiplied, e.g., `prod *= Array[j]`
- `#pragma` the outer for-loop. You could, instead, #pragma the inner for-loop, but your performance would be slower. (Why?)
- Set the running product to 1. inside the outer for-loop, but outside the inner for-loop. This makes it a private variable for each thread.

- Test both static and dynamic scheduling and chunksizes of 1 and 4096. Thus, you will have 4 test cases and a range of thread numbers for each.
- You can control static vs. dynamic scheduling and the chunksize by adding a clause to the end of the #pragma line. Use
  either schedule(static,1) or schedule(static,4096) or schedule(dynamic,1).
  or schedule(dynamic,4096).
- Record the data in units of something that gets larger as speed increases. Joe Parallel used "MegaMults Per Second".

```
// count of how many multiplications were done:
long int numMuled = (long int)ARRAYSIZE * (long int)(ARRAYSIZE+1) / 2;

fprintf( stderr, "Threads = %2d; ChunkSize = %5d; Scheduling=static ;
MegaMults/sec = %10.2lf\n",
        NUMT, CHUNKSIZE, (double)numMuled/(time1-time0)/1000000. );
```

but you can use anything that makes sense.

- Your commentary write-up (turned in as a PDF file) should include:

  1. Tell what machine you ran this on
  2. Create a table with your results.
  3. Draw a graph. The X axis will be the number of threads. The Y axis will be the performance in whatever units you sensibly choose. On the same graph, plot 4 curves:
     - static,1
     - static,4096
     - dynamic,1
     - dynamic,4096
  4. What patterns are you seeing in the speeds?
  5. Why does chunksize 1 vs. 4096 matter like this?
  6. Why does static vs. dynamic matter like this?