Live Lecture #2 Chat Window

April 12, 2023

15:29:48 Sorry, I'm struggling to understand what the performance would be measuring. I know it's time (per second) but what performance? Pins that fit per second?

I would use a measure of how many Monte Carlo trials you can do per second, something like "MegaTrials/Sec".

15:33:07 For the parallel fraction, do we do it for all cores/thread counts we analyze, or just pick one? Or do it for our max cores? I see it says the estimate the parallel fraction, but not sure which core count to use as the reference point(s).

I would do it for the threads/trials combination that gives you the best performance.

15:36:47 What does it mean to estimate the probability?

Probability = (float)numSuccesses / (float)NUMTRIALS;

15:36:57 (1) What strategies would you implement to manage zero div error that produces 'inf' values on fast executions? (2) Why would this happen on 1 & 2 threads but not >= 4 threads? (3) Does that mean the 1 trial is replicated across 4 Threads? (4) What does that mean for accuracy?

I am still mystified why time1 ever == time0 (and thus get a divide by zero). Is this on your own machine? If so, maybe the clock isn't as accurate as one would hope. Have you tried it on flip? Does it get the inf there too?

15:37:25 When I run the shell script on Windows - about 1 in 10 times I get an error message that says "collect2.exe error ld returned 1 exit status". Is it ok if I just run the shell script multiple times and take the cases where it worked? Or is there a way to fix this?

No idea what that message means or why it happens intermittently. Yes, try doing it that way.

15:39:16 I had to change my graph range to logarithmic scale to correctly show the data. is that ok?

It's OK to do a log scale as long as you also do a linear scale. The linear scale will give a truer picture of the shape of the curves.

15:41:34 How shall we make sense of fluctuations in output data? I see a consistent "weird" spot between 1000 and 10000 trials.

Not sure why it would happen consistently in the same place. I can see it if it was happening randomly.

15:43:54 When I do the print _OPENMP to print the version, I get a version date of 200203, which looks like it corresponds to OpenMP v2.0. Are there any features I'm missing relevant to the class that I'll be missing?

No problem. All the features we will be using in OpenMP were around at that time.

15:45:43 There was some chat on Discord about how some of the windows compilers are using different accuracy for the timestamp, for some reason, which can result in two timestamps appearing to be the same, if I remember correctly - so it may be related to that (I had a similar issue when using the Min-GW compiler). Visual Studio's built-in compiler has worked well for me so far.

I would believe that.

15:49:03 With my machine vs flip servers, the shapes were very similar. My processor has 12 threads and when ran through a script, the 12 threads test were the fastest by a significant margin. Is this something that would be expected? or is there something that I should look at. I was also curious about the specs of the flip servers?

That's what I have seen before. The absolute performance numbers might be different, but the shapes of the curves are very similar.

16:03:36 I had a quick question on whether loop variables need to be private - if I write something like this:

```
int i;
#pragma omp parallel for
for (i = 0, i < 10, i++) {
   // procedure
}
...should I declare i as private?
```

In this case, because ii was declared outside the #pragma, i must be declared private(). If you did it this way: for(int i = 0; … then it would automatically have been private without the need to declare it because it was declared after the #pragma..

16:12:37 From Charles Cal to Everyone:
is it always 1 thread per section, or can you assign more threads per section?

It is always one thread per parallel section.

16:14:37 Is there a big difference between "section" and fork()? (Aside from having to say 'if (this_pid) == 0'

OpenMP threads share the same executable, same globals, and the same heap. fork( ) makes a copy of all of that so that what each program sees can diverge wildly. But, your idea is a good one – fork() is how we all tried to do such parallel programming before tools like OpenMP. It didn't work very well.

16:17:13 For the synchronization example, while(omp_test_lock(&Sync) == 0) {…}, when the lock is available, and it skips the while loop code block to execute code below, will it come back and execute the code within the while loop block when the lock becomes unavailable?

Not automatically, but typically we have a bigger loop around that code that would eventually bring it back.

16:20:33 How have you learned to observe opportunities to apply these parallel tools?

Sometime specific projects. Sometimes a hard-to-control curiosity to see what would happen in certain situations.

16:31:25 Is an exception thrown if a section contains a different quantity of WaitBarrier() than another thread?

Probably doesn't throw an exception, probably either hangs or just gets out of sync and your values are wrong.

16:32:46 I've been wondering what happens with errors in general? What happens if you have an error in one thread and all the others are busy waiting for it?

If the error is something serious like a seg fault, that one thread signals all the others to close. If less serious, like a function returns a -1 error code, then life just goes on.