UPPSALA
UNIVERSITET

Assignment 2 Notes

Nima Ghoroubi - nima.ghoroubi@it.uu.se

8 oktober 2019

1 Thread and core

Pay great attention to your choice of words. Thread is by no means a substitute for core or the other way around! They are two very different things. Threads run on cores, and cores take care of the computation. Hyper-threading allows a core to get a better work allocation but it is by no means doubling what hardware you have! Imagine CPU can calculate 10 units of data, but it is only fed 6 units per unit of time. This means the CPU will wait for the next batch while idling, hyperthreading fills that gap by queing the next work so that CPU is never short of data to process. This brings us to the next note.

2 Thread count!

You cannot just spawn arbitrary number of threads aiming for a speedup. There is logic and physics behind spawning threads to gain speedup. For a machine that has 4 physical cores, spawning 40 threads means you have not understood parallelism properly. Weirdly I have seen in the reports of 250, 500, 1000 and even 16348 threads spawned for a script. That is by no means what the assignments ask for! Unless you are running a code on a super computer with 60000 cores, that number of threads mean that you are suffocating the CPU with something that can be done with much less amount of complication. With that to the next note we go!

3 Threads and problem size

Now, just because you have lets say 8 physical and 16 logical cores it does not mean you can spawn 2 - 16 threads and look for a speedup. Speedup is only gained if your CPU is utilized! You cannot bring 2 trucks to move 20 eggs and ask for a visible speedup in the process. You can only expect improvement if your problem is large enough for the parallelism to make sense, other than that you are introducing overhead to a currently menial task, yielding what some have reported back as ”more threads took more time to compute” which is by no means what we are aiming for.

4 Timing

Next point is not unrelated to previous ones, timing is everything! You cannot see any benefit if you are reporting times in the order of milliseconds, then adding 16 threads to the same size. Of course you can report the numbers for small problems, but parallelism makes sense when your runtime is significant enough for parallelism to show a reasonable improvement.

5 Speedup

After considering all of these factors, we come to reporting a logical overview of what we get back from the machine and our code. Speedup plot is the tool. The plot should be as simple and as friendly as possible, take a look at this guide from KTH. If you have to plot multipe speedups in the same figure, use a good legend tagging each line. Each line should persent a simulation you were testing so you cannot just randomly choose what your x and y axis shows. For a better overview of this refer to the KTH tutorial again.

Happy coding,
Nima