PS5 (AMATH 583)

Ben Francis

rnorm

First, let me describe my implementation of rnorm.
rnorm is being fed a value "levels", which corresponds to the number of threads we want to spawn at the base case/bottom level of the recursion.
We also define the base case as 1, not 0.
If we just subtract 1 level each time, and start from 8, and define the base case as 0, the number of threads is like:
Level = 8, Threads = 1
Level = 7, Threads = 2
Level = 6, Threads = 4
Level = 5, Threads = 8 ** We want to stop here!**
Level = 4, Threads = 16
Level = 3, Threads = 32
Level = 2, Threads = 64
Level = 1, Threads = 128
Level = 0, Threads = 256 ** Base case **

Whereas, with my implementation, it is:
Level = 8, Threads = 1
Level = 4, Threads = 2
Level = 2, Threads = 4
Level = 1, Threads = 8 ** Base case, and we want to stop here! **

**** How much parallel speedup do you see for 1, 2, 4, and 8 threads?***

I will examine one token problem size, which roughly characterizes the entire trend.
The problem size looked at is 4194304.

2:1 --> 8.45:5.24 --> 1.61
4:1 --> 8.45:5.24 --> 1.61
8:1 --> 9.27:5.24 --> 1.76

This approximately characterizes the results for the otehr sizes as well; best performance for 8 threads, about 1.6 speedup for 2 and 4.

**** What will happen if you use ``std:::launch::deferred`` instead of ``std:::launch::async`` when launching tasks?***
When will the computations happen? Will you see any speedup? For your convenience, the driver program will also call ``recursive_two_norm_b`` -- which you can implement as a copy of ``recursive_two_norm_a`` but with the launch policy changed.

Theory:
The computations will prep the futures, but wait to run them until the "get" call, and we should see no speedup.
Experiment:
There is no speedup, until the 8 thread case, at which point there is approiximately 1.5X speedup.

My guess for why the speedup in the 8 thread case is that at a certain depth of the recursion,
there is a difference for when the get calls are made, so that they're not made sequentially like we have previously seen in the for loop.
Then the threads will actually run parallel to some degree, and we get improvement.
Seeing this in the 8 thread case seems like evidence for the fact that this effect is more pronounced with deeper levels of recursion, as expected.