CPSC 424/524 Assignment 2 (Spring 2018)

Ezra Davis

Development environment, building and running

```bash
srun --pty -p interactive -c 10 -t 6:00:00 --mem-per-cpu=6100mb bash

module load Langs/Intel/2015_update2

./build.sh
```

Should both build and run the program with all quantities of threads and parallel configurations.

Program output

```
Threads,    Iterations, Time,       Area,       Parallel type
1,      500,         1.768816,   1.513484,   serial
1,      500,         1.787005,   1.513484,   for
1,      500,         1.682113,   1.513484,   for static 1
1,      500,         1.680082,   1.513484,   for static 10
1,      500,         1.734529,   1.513484,   for dynamic
1,      500,         1.678980,   1.513484,   for dynamic 10
1,      500,         1.661257,   1.513484,   for guided
1,      500,         1.687749,   1.513484,   for collapsed
1,      500,         1.854567,   1.513484,   for guided collapsed
1,      500,         1.706955,   1.513484,   task per row
1,      500,         1.705420,   1.513484,   task shared row creation
1,      500,         1.851591,   1.513484,   task
Threads,    Iterations, Time,       Area,       Parallel type
2,      500,         1.680520,   1.513484,   serial
2,      500,         1.458318,   1.513684,   for
2,      500,         0.841882,   1.513620,   for static 1
2,      500,         0.839940,   1.513564,   for static 10
2,      500,         0.841747,   1.513674,   for dynamic
2,      500,         0.839488,   1.513642,   for dynamic 10
2,      500,         0.833248,   1.513508,   for guided
2,      500,         1.422057,   1.513544,   for collapsed
2,      500,         1.428693,   1.513544,   for guided collapsed
```

```
2,      500,         0.890367,   1.513492,   task per row
2,      500,         0.868772,   1.513732,   task shared row creation
2,      500,         1.257136,   1.513838,   task
Threads,    Iterations, Time,       Area,       Parallel type
4,      500,         1.680254,   1.513484,   serial
4,      500,         0.793007,   1.513422,   for
4,      500,         0.420551,   1.513666,   for static 1
4,      500,         0.420648,   1.513746,   for static 10
4,      500,         0.420686,   1.513520,   for dynamic
4,      500,         0.419518,   1.513684,   for dynamic 10
4,      500,         0.419386,   1.513492,   for guided
4,      500,         0.758596,   1.513384,   for collapsed
4,      500,         0.762039,   1.513388,   for guided collapsed
4,      500,         0.489103,   1.513524,   task per row
4,      500,         0.448401,   1.513460,   task shared row creation
4,      500,         1.154752,   1.513390,   task
Threads,    Iterations, Time,       Area,       Parallel type
8,      500,         1.679760,   1.513484,   serial
8,      500,         0.487977,   1.513558,   for
8,      500,         0.210329,   1.513650,   for static 1
8,      500,         0.212236,   1.513690,   for static 10
8,      500,         0.210533,   1.513552,   for dynamic
8,      500,         0.211370,   1.513646,   for dynamic 10
8,      500,         0.208173,   1.513716,   for guided
8,      500,         0.460859,   1.513602,   for collapsed
8,      500,         0.463099,   1.513678,   for guided collapsed
8,      500,         0.353733,   1.513428,   task per row
8,      500,         0.337912,   1.513500,   task shared row creation
8,      500,         5.648783,   1.513474,   task
Threads,    Iterations, Time,       Area,       Parallel type
10,     500,         1.678893,   1.513484,   serial
10,     500,         0.531106,   1.513716,   for
10,     500,         0.194077,   1.513452,   for static 1
10,     500,         0.185972,   1.513580,   for static 10
10,     500,         0.168464,   1.513540,   for dynamic
10,     500,         0.168706,   1.513660,   for dynamic 10
10,     500,         0.166911,   1.513448,   for guided
10,     500,         0.419629,   1.513588,   for collapsed
10,     500,         0.421701,   1.513744,   for guided collapsed
10,     500,         0.350756,   1.513472,   task per row
10,     500,         0.332447,   1.513472,   task shared row creation
10,     500,         8.098166,   1.513408,   task
```

Serial Program

OpenMP Loop Directives

(Please see program output above)

As you can see, the `guided` results followed by `dynamic`, then `dynamic(10)`, and then `static(10)` had the minimum runtimes, beating serial even (oddly) in the single-threaded case.

The `collapse` clause didn't appear to help - in this case, smaller chunks aren't necessarily better for the `guided` for loops. On the other hand, the bare `for` (with default scheduling) loops benefitted from the `collapse` clause.

OpenMP Tasks

(see program output above)

The single cell tasks were disasterous - taking far longer than even the serial cases. This makes sense, because there is a sizeable overhead in making each task. That said, when task creation was shared among multiple threads, it was nearly as fast as the more intelligent for loops, and faster than the for loop with default scheduling. When an individual thread created a task per row, it was reasonably fast, but not quite as fast as the shared task creation method.

Parallel Random Number Generator

I created a very simple method of generating different random numbers: initialize a different seed in each of the threads. I do this by calling `dsrand(12345)` in a parallel region, and adding the thread number to the seed. If we were using a more complex or opaque randomization function (e.g. one that required a more randimized initial seed) or had more stringent randomness requirements, this might not have been sufficient, but as it is, its fine.