Bryce Graves

Professor Mike Bailey

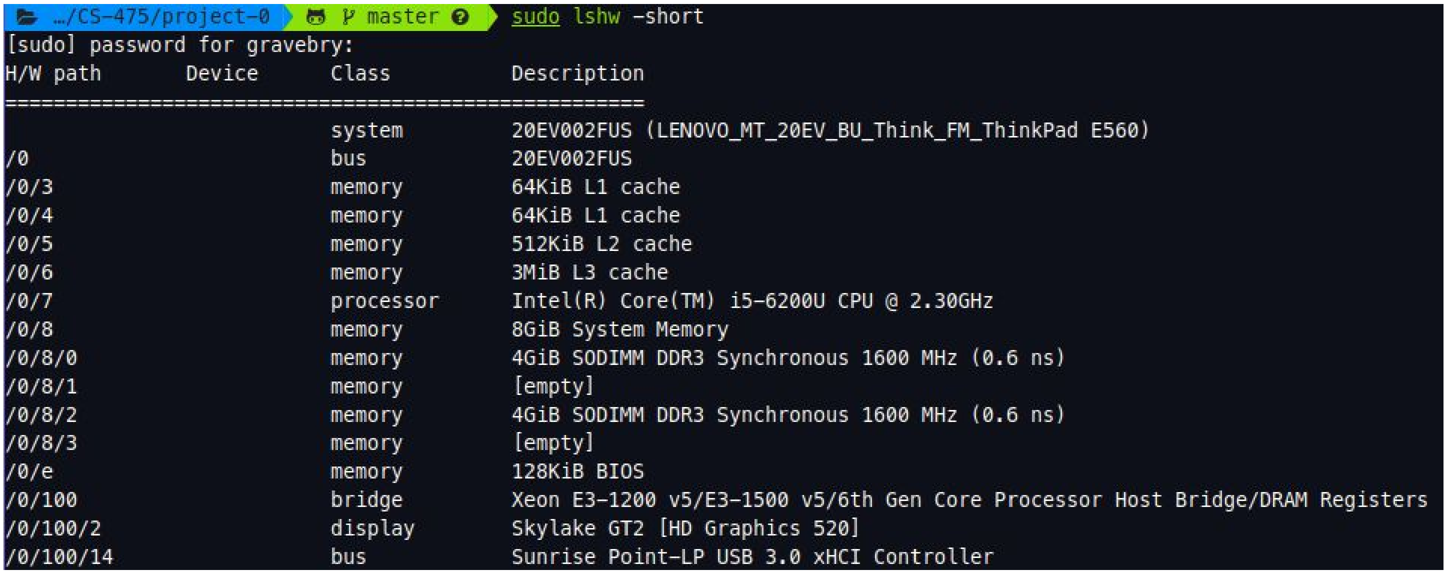
CS 475

2020 4 6 (ISO 8601)

Project: 0

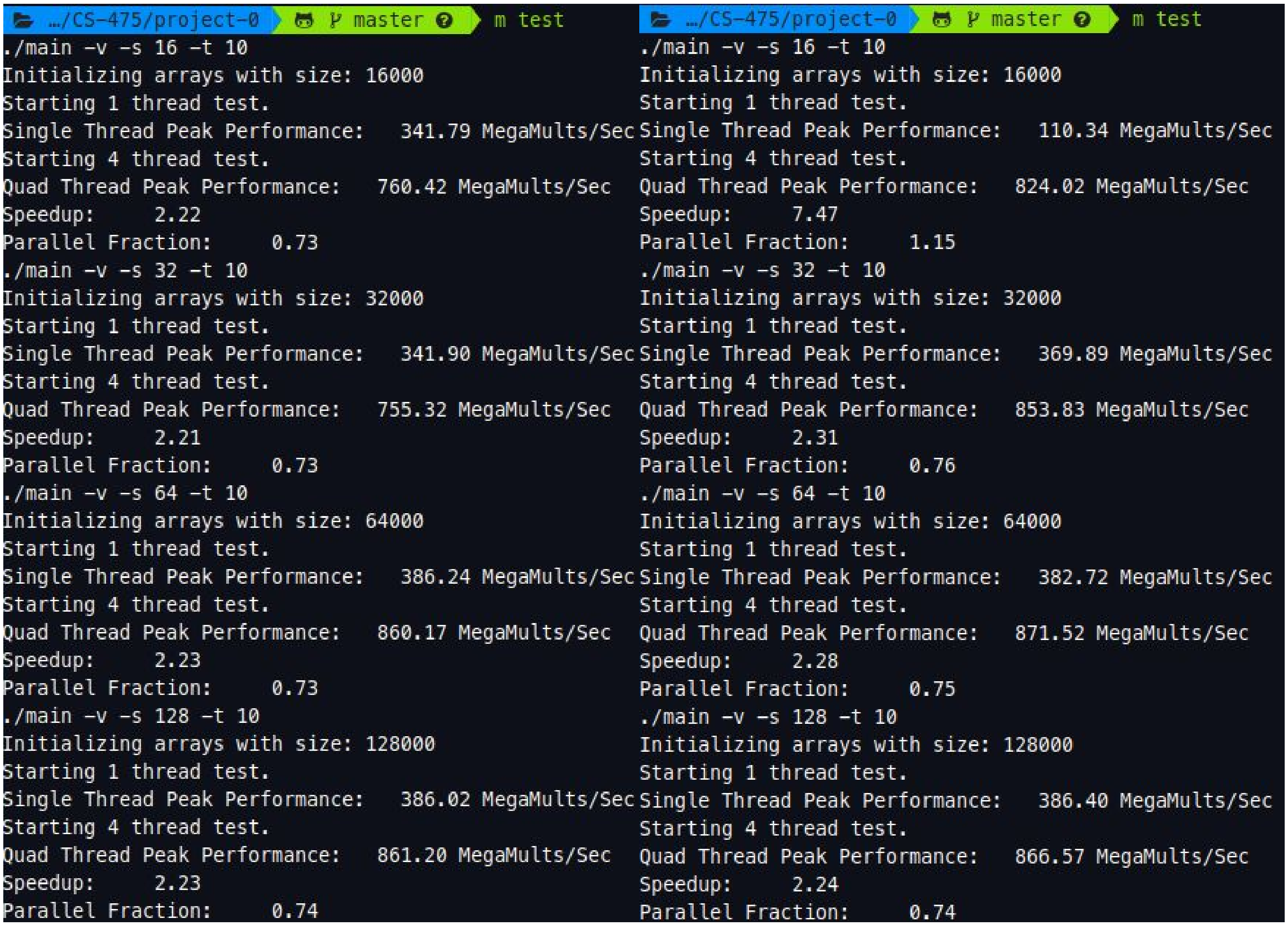
Bryce Graves
gravebry@oregonstate.edu

System Hardware: personal laptop details



```
 …/CS-475/project-0   master   sudo lshw -short
[sudo] password for gravebry:
H/W path       Device      Class          Description
=====================================================
                           system         20EV002FUS (LENOVO_MT_20EV_BU_Think_FM_ThinkPad E560)
/0                         bus            20EV002FUS
/0/3                       memory         64KiB L1 cache
/0/4                       memory         64KiB L1 cache
/0/5                       memory         512KiB L2 cache
/0/6                       memory         3MiB L3 cache
/0/7                       processor      Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz
/0/8                       memory         8GiB System Memory
/0/8/0                     memory         4GiB SODIMM DDR3 Synchronous 1600 MHz (0.6 ns)
/0/8/1                     memory         [empty]
/0/8/2                     memory         4GiB SODIMM DDR3 Synchronous 1600 MHz (0.6 ns)
/0/8/3                     memory         [empty]
/0/e                       memory         128KiB BIOS
/0/100                     bridge         Xeon E3-1200 v5/E3-1500 v5/6th Gen Core Processor Host Bridge/DRAM Registers
/0/100/2                   display        Skylake GT2 [HD Graphics 520]
/0/100/14                  bus            Sunrise Point-LP USB 3.0 xHCI Controller
```

Testing output:



```
…/CS-475/project-0  master  m test
./main -v -s 16 -t 10
Initializing arrays with size: 16000
Starting 1 thread test.
Single Thread Peak Performance:   341.79 MegaMults/Sec
Starting 4 thread test.
Quad Thread Peak Performance:   760.42 MegaMults/Sec
Speedup:     2.22
Parallel Fraction:     0.73
./main -v -s 32 -t 10
Initializing arrays with size: 32000
Starting 1 thread test.
Single Thread Peak Performance:   341.90 MegaMults/Sec
Starting 4 thread test.
Quad Thread Peak Performance:   755.32 MegaMults/Sec
Speedup:     2.21
Parallel Fraction:     0.73
./main -v -s 64 -t 10
Initializing arrays with size: 64000
Starting 1 thread test.
Single Thread Peak Performance:   386.24 MegaMults/Sec
Starting 4 thread test.
Quad Thread Peak Performance:   860.17 MegaMults/Sec
Speedup:     2.23
Parallel Fraction:     0.73
./main -v -s 128 -t 10
Initializing arrays with size: 128000
Starting 1 thread test.
Single Thread Peak Performance:   386.02 MegaMults/Sec
Starting 4 thread test.
Quad Thread Peak Performance:   861.20 MegaMults/Sec
Speedup:     2.23
Parallel Fraction:     0.74
```

```
…/CS-475/project-0  master  m test
./main -v -s 16 -t 10
Initializing arrays with size: 16000
Starting 1 thread test.
Single Thread Peak Performance:   110.34 MegaMults/Sec
Starting 4 thread test.
Quad Thread Peak Performance:   824.02 MegaMults/Sec
Speedup:     7.47
Parallel Fraction:     1.15
./main -v -s 32 -t 10
Initializing arrays with size: 32000
Starting 1 thread test.
Single Thread Peak Performance:   369.89 MegaMults/Sec
Starting 4 thread test.
Quad Thread Peak Performance:   853.83 MegaMults/Sec
Speedup:     2.31
Parallel Fraction:     0.76
./main -v -s 64 -t 10
Initializing arrays with size: 64000
Starting 1 thread test.
Single Thread Peak Performance:   382.72 MegaMults/Sec
Starting 4 thread test.
Quad Thread Peak Performance:   871.52 MegaMults/Sec
Speedup:     2.28
Parallel Fraction:     0.75
./main -v -s 128 -t 10
Initializing arrays with size: 128000
Starting 1 thread test.
Single Thread Peak Performance:   386.40 MegaMults/Sec
Starting 4 thread test.
Quad Thread Peak Performance:   866.57 MegaMults/Sec
Speedup:     2.24
Parallel Fraction:     0.74
```

Analysis:

I added optarg parsing to the given program to enable setting the array size (based on the number multiplied by a thousand), number of test runs before using the fastest result, and a toggleable verbose option. Above are the results of just two of the test sets I ran. Overall the parallel fraction seemed to tend towards 0.95 and the speedup towards 3.5 over the course of testing. Early on I had slower second run results due to not resetting the arrays to their original values. But after ensuring that the arrays are the same for both, I observed an increase of the parallel fraction and speedup from 0.75 to 0.95 and 2.5 to 3.5 respectively. These results being below the projected 4.0 speedup is expected due to the overhead of parallelization.