**CMPT 431**

**Assignment 2 Report**

**Adam Penner**

All of the following tests were conducted on a machine with the following specs:

Architecture: x86\_64

CPU op-mode(s): 32-bit, 64-bit

Byte Order: Little Endian

CPU(s): 8

On-line CPU(s) list: 0-7

Thread(s) per core: 2

Core(s) per socket: 4

Socket(s): 1

NUMA node(s): 1

Vendor ID: AuthenticAMD

CPU family: 21

Model: 2

Model name: AMD FX(tm)-8350 Eight-Core Processor

Stepping: 0

CPU MHz: 1400.000

CPU max MHz: 4000.0000

CPU min MHz: 1400.0000

BogoMIPS: 7982.17

Virtualization: AMD-V

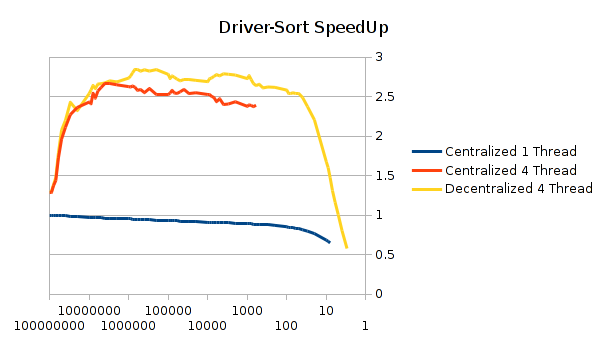
L1d cache: 16K

L1i cache: 64K

L2 cache: 2048K

L3 cache: 8192K

NUMA node0 CPU(s): 0-7

****

Part 1)

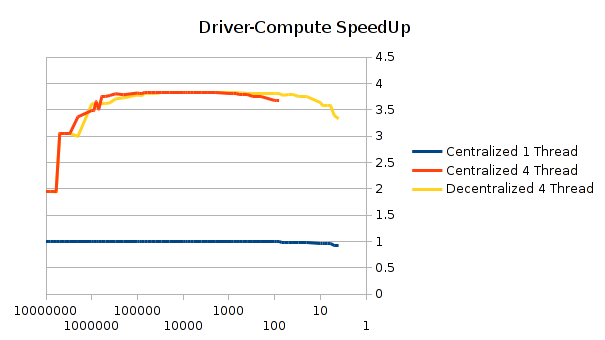
The smallest grainsize that is still within 5% of the infinite grainsize is a grainsize of 700,000 (with an infinite grainsize of 100,000,000). The infinite grainsize time was 6.08371 seconds while the other grainsize was 6.39825 seconds. As the assignment stated, the reason why using the particle size as the grainsize is due to the fact that the code simply acts on the sorting algorithm then. But once we start splitting into smaller grainsizes, there is added overhead from the action of splitting the array.

Part 2)

The maximum speedup on four cores is 2.8585x when comparing the grainsize of 20,000 between single core and four cores.

Part 3)

For my tests the driver-sort code, the distributed queuing was always quicker than the centralized version (other than a single outlier). The distributed queue helps reduce contention as it allows for more tasks per thread to be done before attempting to steal from another thread. Once the stealing starts though, it could also be distributed across multiple threads which could cause for a lesser rate of contention. With the distributed queuing though, there is a balance between thread work load and granularity. Both curves, especially the decentralized version, shows this balance perfectly. With the higher grainsize, we have a load imbalance dominates as there is more work to do, while on the smaller grainsize we have increased overhead due to enqueuing and dequeuing many tasks.



Part 1)

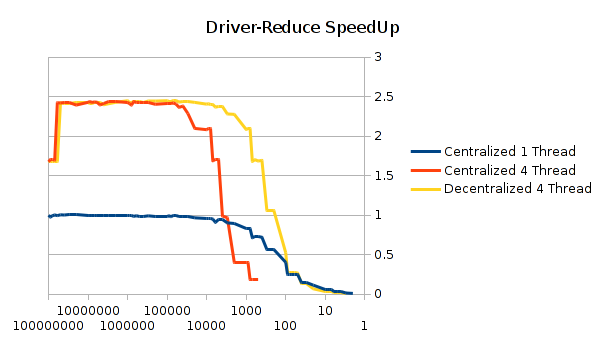
The smallest grainsize that is still within 5% of the infinite grainsize is a grainsize of 6 (with an infinite grainsize of 10,000,000). The infinite grainsize took 7.71435 seconds while the other took 8.04149 seconds.

Part 2)

The maximum speedup on four cores was 3.8364x compared to the single core. With the grainsize of 9000, the four cores took 2.01337 seconds to compute and 7.72407 seconds for the respective single core.

Part 3)

The distributed queue was not always the fastest. In the case of centralized and distributed queues, there was barely any difference between the times. The major difference that was noted is that with the distributed queuing system we are able to use smaller grainsizes before hitting a high overhead due to queuing and dequeuing tasks. There was also a small blip in between 100K and 1M grainsize where the distributed was actually worse than the centralized.



Part 1)

The smallest grainsize that is still within 5% of the infinite grainsize is a grainsize of 7,000 (with an infinite grainsize of 100,000,000). The infinite grainsize took 0.1184 seconds while the other took 0.1245 seconds.

Part 2)

The maximum speedup on four cores was 2.4709x compared to the single core. With the grainsize of 400000, the four cores took 0.0487 seconds and 0.1204 seconds for the respective single core.

Part 3)

The queuing was not always the fastest in this case. For grainsizes equal or greater than a million, both distributed and centralized queuing systems were almost the same. Less than a million though, centralized queuing slowly deteriorated and then explosively became worse at about the 5000 grainsize mark. Ultimately the decentralized had the same downfall in performance, but the because of the distributed queuing it allowed us use a smaller grainsize before succumbing to enqueuing and dequeuing overheads.