Maxwell Daum

12/12/2016

Comp 633 PA2

I chose to implement an in-place parallelized quicksort using OPENMP 3.0 Tasks on BASS and Phaedra. Quicksort does not lend itself well to being parallelized due to the bottleneck of available parallelism at the beginning of the algorithm. Since the partition function is sequential, N sequential work is done every time. Not only that, if you are running with say 16 threads, after the first two tasks are spawned, you are only using 2 of the 16 processors, then 4 and so on. So the more processors you throw at the problem, the longer it takes for full available parallelism to be achieved. Nevertheless, modest speedup can be achieved by simply using tasking.

The first hurdle the project provides (beyond parallelizing the code) was to find a problem size range such that the overhead of thread spawning and task management did not eclipse the computation time. After much trial and error, I settled on a lower bound of 1 Million doubles, and scaling in 20 Million increments all the way to 100 Million doubles. This range showed the adverse effects of parallel tasking on small problems in the low range, and quickly outshined the sequential implementation on sizes greater than 1 Million.

The next task was to try and lower the amount of thread spawning when the current sub problem size was small enough that it would be better to do sequentially. Looking at BASS L1 and L2 cache sizes (64KB and 1MB respectively), I chose 133000 doubles to be the sequential threshold. This provided a good boost in efficiency.

On Bass and Phaedra I ran 10 runs on every combination of problem size and number of processors. Below are my sorting performance and speedup results:

On Bass, 16 processors yields a modest 7 time speedup. As mentioned before, this is due to the beginning divide and conquer levels of the algorithm when there is a significant amount of work and limited parallelism. Comparing this with the speedup of approximately 2 times when running with 2 processors validates this sentiment. At problem size 1 Million, speedup is dramatically reduced due to the overhead of tasking.

On Phaedra, the sequential speed compared to that of BASS is impressive. Speedups are not as attractive as BASS, with only a 7.3 speedup with 20 processors. I would think that the extremely efficient sequential runs are contributing factors to this, and better results may come from even larger problem sizes.

In order to validate the bottleneck of upfront low available parallelism, I did a running time analysis on each level of the divide and conquer “tree”. By capturing the earliest time each level is worked on and the latest time each level is finished, I obtained a timeline of the run (albeit slowed down slightly due to needing to make critical updates to bookkeep this information). Below is a dump of a timeline for BASS with 16 processors, and Phaedra with 20 processors respectively on problem size 100 Million:

(I have shaved off the non-important numbers for an easier read)

BASS

level 0: start:93.672603 end:94.726329 time taken:1.053726

level 1: start:94.726333 end:95.869545 time taken:1.143212

level 2: start:94.987163 end:96.353579 time taken:1.366415

level 3: start:95.097853 end:96.814651 time taken:1.716797

level 4: start:95.133995 end:97.078640 time taken:1.944645

level 5: start:95.140360 end:97.373917 time taken:2.233557

level 6: start:95.149049 end:97.608574 time taken:2.459526

level 7: start:95.159377 end:97.990064 time taken:2.830687

level 8: start:95.162706 end:98.196581 time taken:3.033875

level 9: start:95.167610 end:98.356024 time taken:3.188414

level 10: start:95.172943 end:98.442295 time taken:3.269352

level 11: start:95.219586 end:98.512195 time taken:3.292609

level 12: start:95.227118 end:98.567875 time taken:3.340758

level 13: start:95.236790 end:98.611747 time taken:3.374957

level 14: start:95.308615 end:98.646409 time taken:3.337794

level 15: start:95.329811 end:98.672660 time taken:3.342850

level 16: start:95.520691 end:98.703980 time taken:3.183289

level 17: start:95.560818 end:98.733113 time taken:3.172295

level 18: start:95.565616 end:98.756608 time taken:3.190993

level 19: start:97.048241 end:98.778618 time taken:1.730378

level 20: start:97.085128 end:98.786001 time taken:1.700873

level 21: start:97.787882 end:98.807762 time taken:1.019880

level 22: start:97.811375 end:98.805459 time taken:0.994084

level 23: start:97.836457 end:98.798336 time taken:0.961879

level 24: start:98.553559 end:98.805617 time taken:0.252058

Total Time Taken: 5.135606

PHAEDRA

level 0: start:45.209073 end:45.739605 time taken:0.530532

level 1: start:45.739608 end:46.460372 time taken:0.720764

level 2: start:45.834848 end:46.778419 time taken:0.943571

level 3: start:45.866754 end:47.179015 time taken:1.312261

level 4: start:45.878456 end:47.458163 time taken:1.579707

level 5: start:45.883623 end:47.715771 time taken:1.832148

level 6: start:45.884817 end:47.920105 time taken:2.035288

level 7: start:45.886141 end:48.195533 time taken:2.309392

level 8: start:45.890228 end:48.397291 time taken:2.507063

level 9: start:45.893241 end:48.509988 time taken:2.616747

level 10: start:45.918556 end:48.579212 time taken:2.660655

level 11: start:45.921749 end:48.630549 time taken:2.708800

level 12: start:46.011243 end:48.724915 time taken:2.713672

level 13: start:46.061510 end:48.797967 time taken:2.736457

level 14: start:46.087293 end:48.853147 time taken:2.765854

level 15: start:46.988286 end:48.932667 time taken:1.944381

level 16: start:47.105898 end:48.982824 time taken:1.876927

level 17: start:47.177620 end:49.032395 time taken:1.854776

level 18: start:47.211249 end:49.056790 time taken:1.845541

level 19: start:47.222220 end:49.078958 time taken:1.856737

level 20: start:47.316353 end:49.097063 time taken:1.780710

level 21: start:47.444565 end:49.107942 time taken:1.663377

level 22: start:47.445292 end:49.119955 time taken:1.674663

level 23: start:47.446673 end:49.126998 time taken:1.680325

level 24: start:47.452820 end:49.145375 time taken:1.692556

level 25: start:47.454432 end:49.153720 time taken:1.699289

level 26: start:47.456859 end:49.149607 time taken:1.692748

level 27: start:48.716580 end:49.161231 time taken:0.444651

level 28: start:48.718336 end:49.158247 time taken:0.439911

level 29: start:49.111966 end:49.121270 time taken:0.009305

level 30: start:49.113580 end:49.135654 time taken:0.022074

Total time taken: 3.952234

As expected, the problem starts off with no interleaving start and end times between levels. But starting between level 1 and 2 you notice that level 2’s start time will be before level 1’s end time, indicating the utilization of available parallelism. The interleaving becomes the most pronounced in the middle of the tree, due to the large amount of work and full utilization of available parallelism. I believe these timelines confirm that the initial large partitions with little available parallelism are a huge bottleneck in speedup.

Next steps in improving quicksort would be to find ways to parallelize the beginning sections of work. Unfortunately, I did not allocate enough time do attempt this, but I believe my analysis has sufficiently identified the targets for future optimizations.

Notes on submitted files:

* qsort\_seq.c is the sequential program
* qsort\_par\_v1.c is the parallelized quicksort, with params {problems size, sequential\_threshold, num\_processors}
* qsort\_par\_v2.c is parallelized quicksort, except I hard code the sequential\_threshold to be n/p (not used in data capture)
* qsort\_par\_timing.c is parallelized quicksort but with times being recorded for each level of the divide and conquer “tree”
* time16.txt is the timeline output for BASS
* time20phaedra.txt is the timeline output for Phaedra
* v1data.csv and v1data.xlsx is my data for BASS
* v1phaedra.csv and v1phaedra.xlsx is my data for Phaedra
* Included a makefile that will compile all c files…includes a make clean directive