Erik Bauer ebauer@kth.se

Homework 2

Matrix sum

I was not able to compile my implementation of the parallelized matrix sum program on the KTH linux-shell servers since I use some functionality of openMP that is not supported by the compiler on the servers (user-defined reductions). I have therefore had to run them on my laptop running on a processor with two cores and 4 threads. The program was run five times for each setting and the median execution time was recorded.

Execution times	1 thread	2 threads	4 threads
100^2 (Size of matrix)	0.000187216 s	0.000269028 s	0.00238069 s
10 000^2	0.95367 s	0.48094 s	0.462822 s
1 000 000^2	0.953566 s	0.481134 s	0.464508 s

Speedup	2 threads	4 threads
100	0.695	0.0786
10 000	1.983	2.061
1 000 000	1.982	2.053

When running the program in parallel with a small matrix, it's clear that the parallelization is slowing the program down. The difference to running on larger matrices is quite significant, and I would wager that it it caused by the overhead of creating threads and synchronizing them, taking up a larger amount of the execution time when running on smaller tasks than on larger.

Quicksort

I was able to run my quicksort on the KTH linux servers.

Execution times	1 thread	2 threads	4 threads	8 threads
100 (Size of matrix)	0.0000797049	0.000328565	0.000654344	0.001374439
10 000	0.000077677	0.000312159	0.000688051	0.00136307

1 000 000	0.0000797519	0.000346689	0.000657241	0.00132977

Speedup	2 threads	4 threads	8 threads
100	0.243	0.121	0.058
10 000	0.249	0.113	0.057
1 000 000	0.23	0.121	0.599

Here there was a significant slowdown when using more threads. Could it be that synchronization between threads is making it slow down, perhaps?