Parallel algorithms and parallel computers (in4026)

Assignment C

Sverre Rabbelier
srabbelier@gmail.com
1308211

Introduction

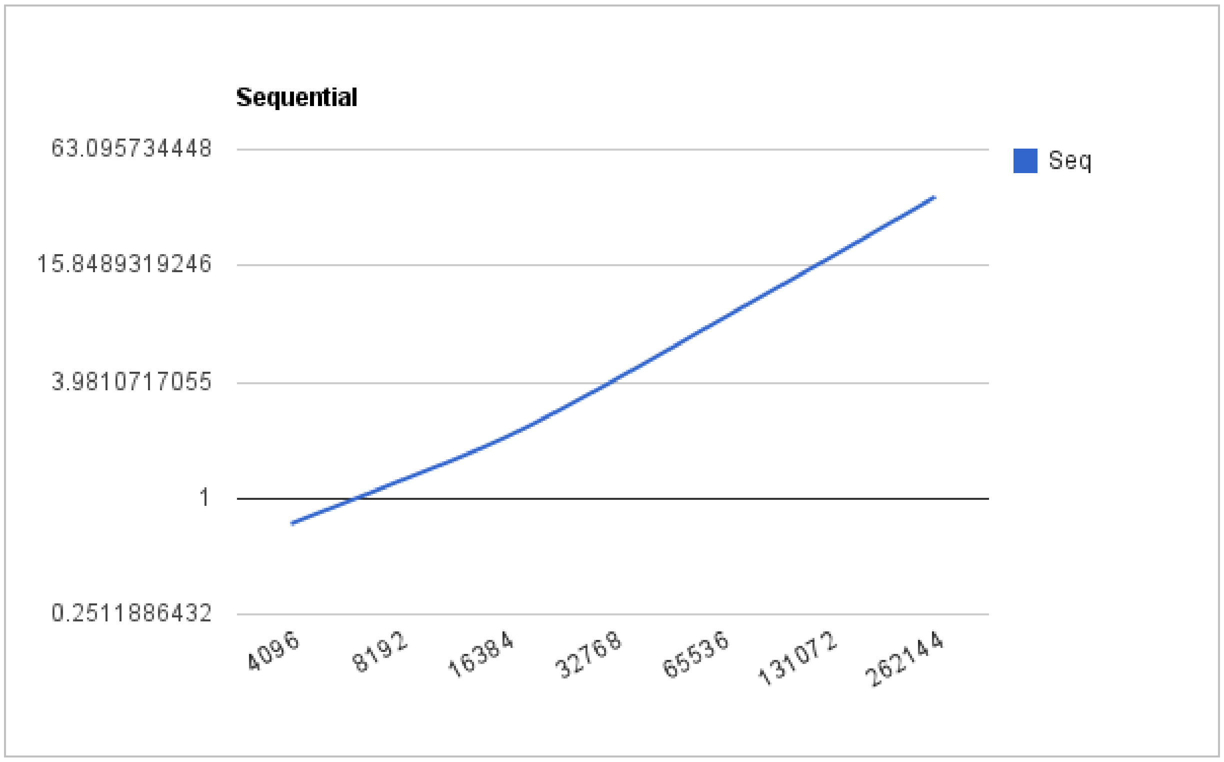
Asked is to provide an algorithm that, given a linked list (represented as an array) to find the distance from one pointer to the last pointer in the list.
My algorithm does this simply by jumping one pointer log(n) times (for all pointers), and recording the distance (increase) for each pointer with each jump.

Time-complexity analysis

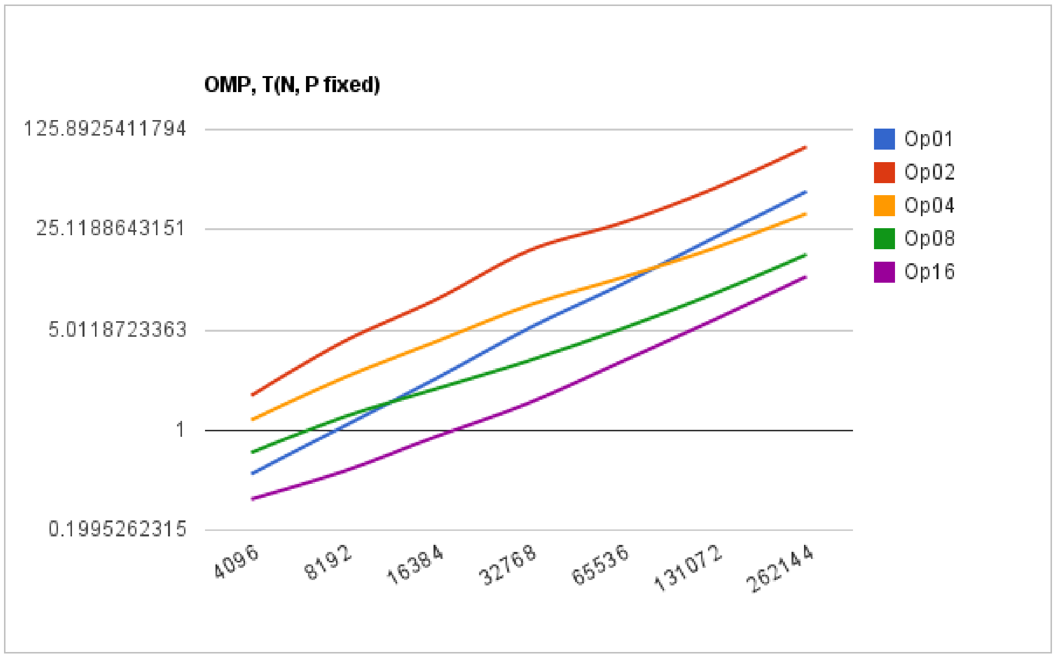
The algorithm requires an initialization which has a O(1, n) work-time complexity.
The actual calculation consists of a loop that is repeated log(n) times. The inner loop has a O(1, n) complexity, resulting in a O(log(n), n*log(n)) work-time complexity

Testing

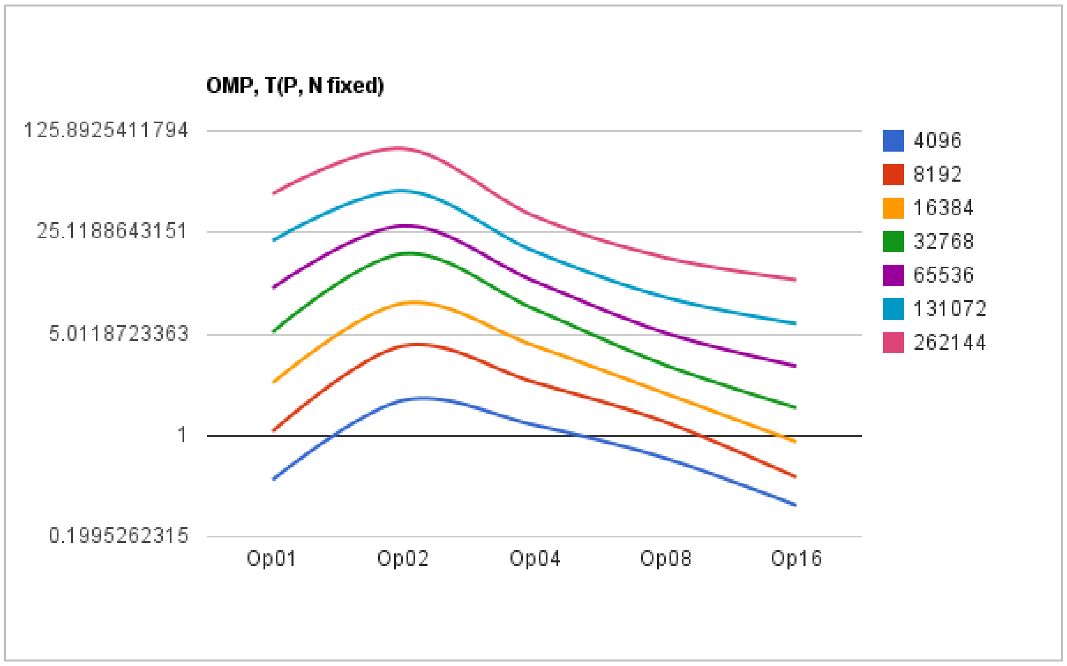
First I tested plotted the linear set to get a baseline. I chose a log scale because that works nicely with the exponentially increasing input size.



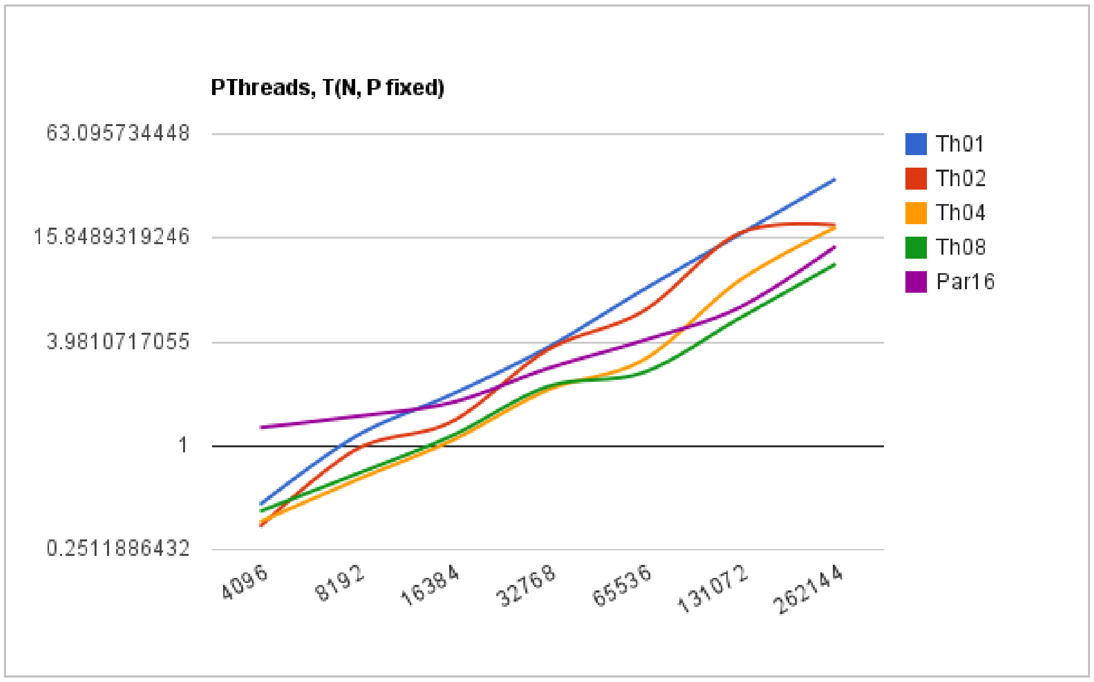
Now I'd expect the OMP to show something similar but of course with lower values for a higher thread count. The single-threaded version seems to be performing a lot better than it should be (being entirely _below_ the nt=2 line).



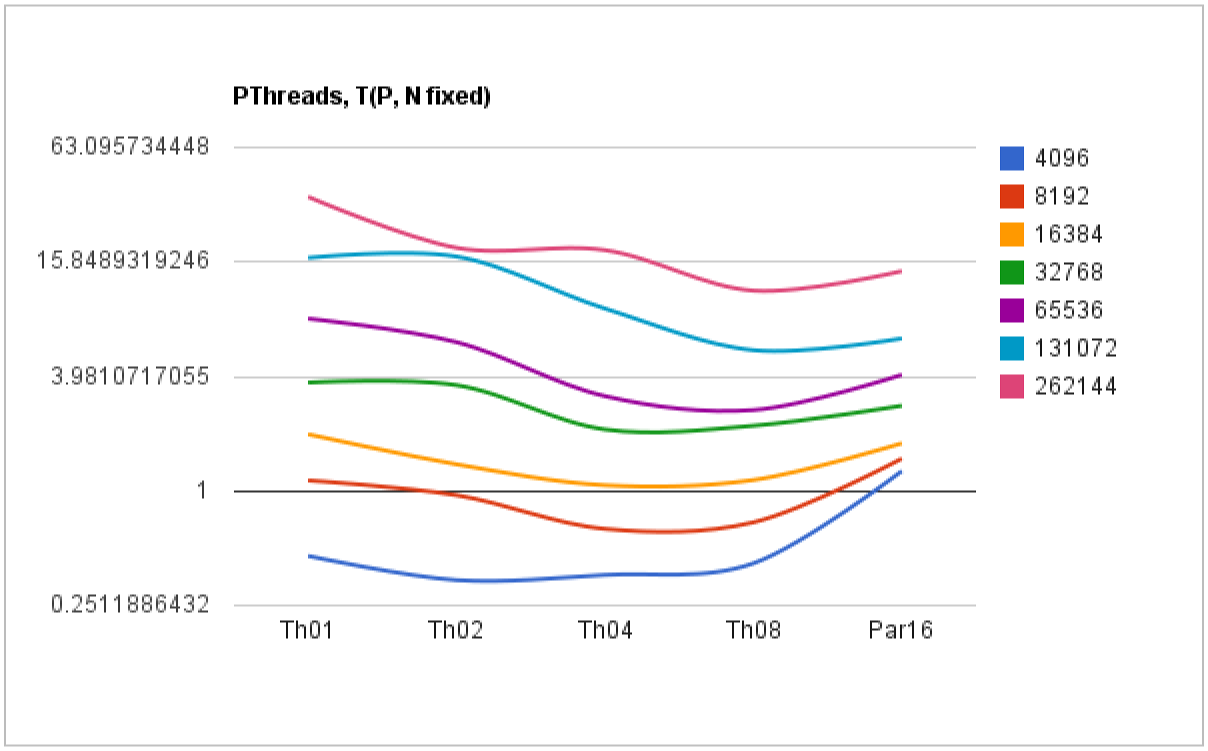
If we instead plot the threads against N, the overhead of running more than one thread is clearly visible.



For pthreads the results are not quite as screwy as in assignment A, but the lines are a lot more wavy than I would like.



Keeping n fixed but varying the number of threads again shows the erraticness of the low N.



Conclusions

Translating this assignment to OpenMP and pthreads turned out to be relatively trivial. Simply adding the appropriate pragma’s and barrier waits took but a few tries to get right. I suspect that if I had increased TIMES further and started out with a higher N that the results would have been more reliable, but as it already takes quite a while to run the algorithm that wasn’t really feasible.