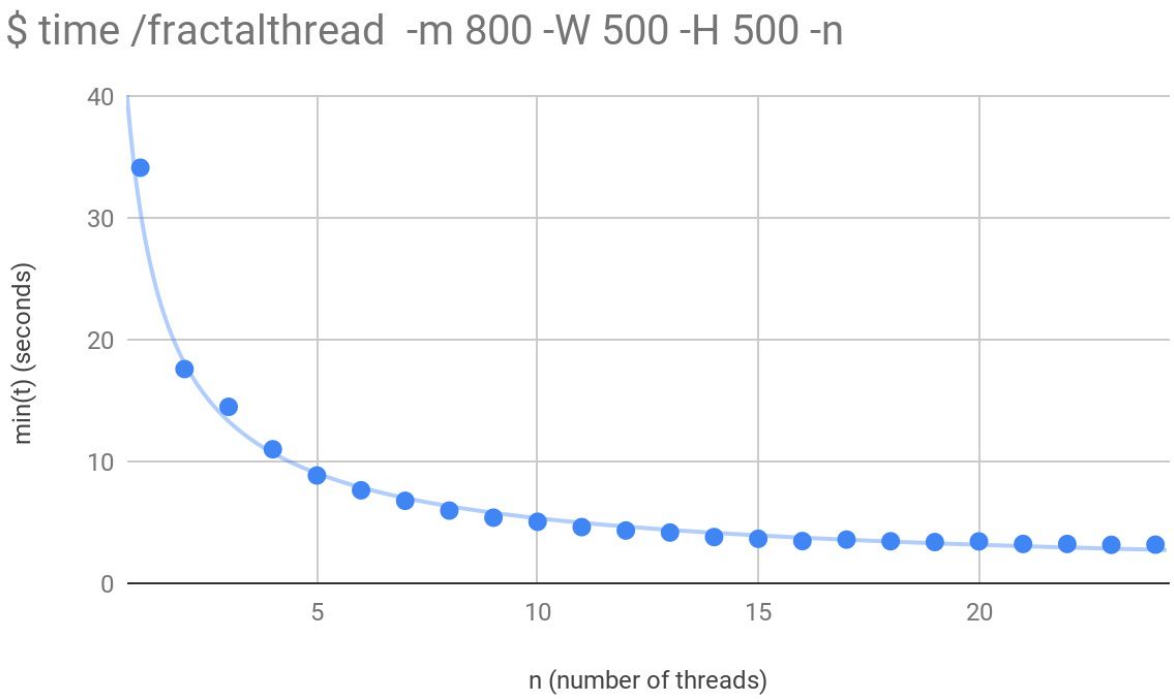
Project III: Parallel Programming Lab Report

Prof. Douglas Thain/ CSE 30341: Operating Systems Principles

Feb 22, 2019 Yifan Yu

1. Run fractalthread on a range of 1-24 threads, and record the total wall clock time. The execution time of these experiments may be upset by other things going on in the machine. So, repeat each measurement five times, and use the fastest time achieved. Plot the results on a graph.



2. What is the optimal number of threads for fractalthread? Explain the significance of this number.

   The optimal number of threads was n = 23. min(t) = 3.154s

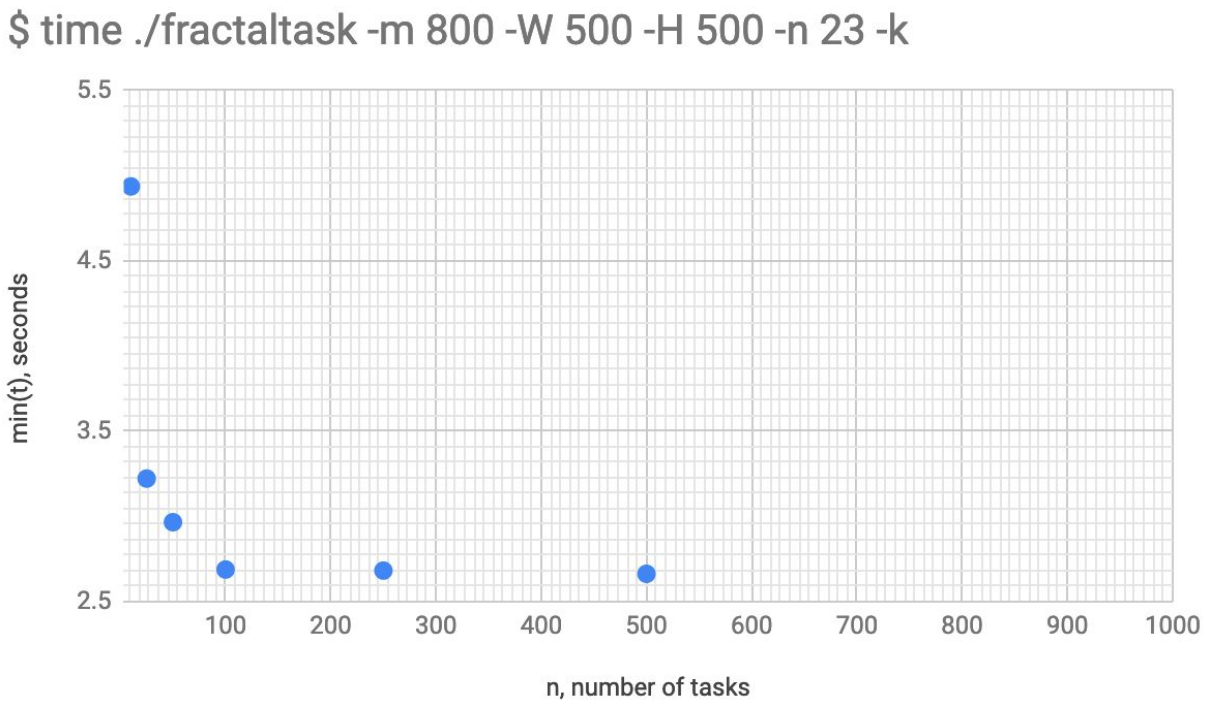
   This number is the most optimal number of threads to maximize the efficiency of the program given the parameters. Given the machine, this number is compatible with the number of cores for optimal efficiency.

3. Is there a downside to having too many threads? Explain why.

   There is a downside to having too many threads. There has to be context switching between the threads and generating and assigning functions to each thread takes time. Joining all the tasks

done by the treads is also time consuming. Therefore, when each task takes shorter than the total assigning and joining of threads, there is a downside to having too many threads. It is also dependent on multiple processors or CPU cores. If there are too many threads, it could exceed the number of threads that the cores and CPU cycles can manage.

4. Using the ideal number of threads from the previous step, run fractaltask with a wide range of *tasks*. As before, repeat measurements and take the fastest. Plot the results on a graph.



5. Explain the shape of the graph obtained. Is it possible to have too few or too many tasks, relative to the number of threads? Explain why.

It seems that there is no obvious disadvantage to having too many tasks, as tasks are stored in a linked list and threads take tasks off of the list not simultaneously

However, there is a disadvantage to having too few tasks, as each task are bigger and having more threads than tasks force threads to wait for nonexistent tasks(when not enough task were assigned). The number of tasks should not be few than the number of threads and the number of cores as it leaves empty threads.