6.895 Final Project Proposal: Parallelizing a Sorting Algorithm (or possibly max-flow)

Paul Youn

October 13, 2003

I propose to parallelize sorting, probably quick sort. My first exposure to parallelizing sorting was from an earlier lecture in this course. I will explore several different approaches to this objective, similar to the approach taken in the first lab. I will first implement the fastest serial implmentation I can achieve for sorting, and then move on to parallelization. I hope to find several ways to speed up the algorithm through parallelization, and plan on researching the topic extensively. Below I have listed some references. I will also carefully analyze the running time of the algorithm I implement, and possibly explore some related algorithms such as a non-random version of quick sort (although the literature I've read indicates that the deterministic version of quick sort runs a constant factor slower than the randomized version of insertion sort in expected run time).

Here is a more formal listing of the goals I hope to achieve. I first plan on researching the topic to attempt to verify that the scope of the problem is appropriate as outlined below. Second, as mentioned above, I will implement a fast serial implementation of quick sort. Third, I will implement a basic parallel version of quick sort. Fourth, I will carefully analyze the runtime of the algorithm. Fifth, I will verify the runtime analysis with empirical testing. Sixth, I hope to find several speedups on the basic parallel implementation.

I think this is an appropriate project for a single person in a 12 unit course. I feel the scope of this project can be considered on par with Lab 1, but will be accomplished by one person instead of two. In addition, while for Lab 1, several known possible speedups weren't implemented, I plan on leaving no stone unturned so to speak. The sixth goal outlined above is where I hope to spend a significant portion of my time. If the previous 4 goals are very easy, I can spend a lot of time searching for speedups, while if the other goals are more difficult than I anticipated, I can spend less time searching for speedups.

Lastly, if this project seems to be too small in scope, I am also prepared to investigate a different algorithm such as a solution to the max-flow problem. My first exposure to this problem was in the class "Network Optimizations" but I haven't explored parallel solutions to the problem. I have been told by Sean Lie, who has previously looked into parallel solutions to the problem, that this problem is significantly harder than a sorting algorithm, so depending on how small the quick sort problem is (after some amount of investigation, but hopefully before significant implementation occurs) I will switch over to the max-flow problem. Any feedback on the scope of either of these projects would be greatly appreciated.

References

[1] Jimenez-Gonzales, et. al. Exploiting the Memory Hiearchy in Parallel Sorting Algorithms. 2000.

[2] Daniel Jimenez-Gonzales, et. al. The Effect Of Local Sort on Parallel Sorting Algorithms. 2002.

[3] Richard P. Brent and Andrew Tridgell. A Fast, Storage-Efficient Parallel Sorting Algorithm. http://web.comlab.ox.ac.uk/oucl/work/richard.brent/pd/rpb140.pdf. 1993.

[4] Scandal Project. Review of Sorting Algorithms. http://www-2.cs.cmu.edu/ scandal/nesl/algorithms.html

[5] 6.895 Lecture Notes. 2003.

[6] A. V. Goldberg and R. E. Tarjan. A New Approach to the Maximum Flow Problem. 1986.

[7] R. Anderson and J. C. Setubal. A Parallel Implementation of the Push-Relabel Algorithm for the Maximum Flow Problem. 1995.