Parallel Sorting of Roughly-Sorted Sequences

CSCI 5172 Fall '16 Project

Anthony Pfaff, Jason Treadwell

September 23, 2016

Sorting a collection according to some ordering among its items is among the most classic problems of computer science. A well-established result is the linearithmic (i.e. $O(n \lg n)$) optimal upper bound for sorting sequences of length n by comparison. Commonly-used sorting algorithms such as Mergesort and Quicksort have $\Theta(n \lg n)$ and $\Omega(n \lg n)$ runtimes, respectively.

We can exploit the ordering of *roughly-sorted* sequences to sort them in $O(n \lg k)$ time, where k is the *radius* of a sequence S or the smallest k such that S is k-*sorted.*[2] A k-sorted sequence $\{a_0, a_1, \cdots, a_n\}$ satisfies $a_i \leq a_j \ \forall \ 1 \leq i \leq j \leq n, i \leq j - k$. Since an unsorted sequence can be at most n-sorted, the worst-case runtime of this algorithm has complexity $O(n \lg n)$.

We intend to study and implement sorting of roughly-sorted sequences using both sequential and parallel[1] versions of the algorithm. In particular, I shall investigate the applicability of the method over very large sequences, accelerated using graphics

hardware. A GPU is dedicated graphics hardware that also excels at performing frequent, basic computations over large amounts of data. The use of GPUs for numerical applications is a recent and rapidly-growing development in computer science and software engineering. Suitable hardware is now commonly available in consumer devices and may herald a paradigm shift in how we efficiently perform common and indispensable tasks like sorting.

I propose to develop implementations of the method using C and Nvidia's CUDA framework, comparing the results with sequential implementations running on both the GPU and CPU as well as with the classic Mergesort and Quicksort algorithms. If time permits, I would also like to compare the method to parallel versions of the latter two algorithms.

References

[1] Tom Altman and Bogdan Chlebus. Sorting roughly sorted sequences in parallel. *Information Processing Letters*, 33(6), February 1990.

[2] Tom Altman and Yoshihide Igarashi. Roughly sorting: Sequential and parallel approach. *Journal of Information Processing*, 12(2), 1989.

[3] J. Cheng, M. Grossman, and T. McKercher. *Professional CUDA C Programming*. Wrox, 1st edition, September 2014.

[4] Intel. *Intel Digital Random Generator (DRNG) Software Implementation Guide*, 1.1 edition, August 2012.

[5] Donald E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, 2nd edition, 1998.

[6] George Marsaglia. Xorshift rngs. *Journal of Statistical Software*, 8(14), July 2003.

[7] N. Wilt. *The CUDA Handbook: A Comprehensive Guide to GPU Programming*. Addison-Wesley Professional, 1st edition, June 2013.