Extending the Streaming Model: Sorting and Streaming Networks

Matthias Ruhl, MIT
Gagan Aggarwal, Stanford
Mayur Datar, Stanford
Sridhar Rajagopalan, IBM Almaden

The need to deal with massive data sets in many practical applications has led to a growing interest in computational models appropriate for large inputs. One such model is "streaming computations" [MP80, AMS99, HRR99], where inputs are provided as a long sequence of data items. In this model, functions are computed by a machine with small local memory making one or a small number of linear passes on the input stream. In this talk, motivated by practical considerations, we discuss two extensions of this computational model. Despite being quite different in motivation and form, these extensions turn out to be closely related in their computational power. This suggests that the computational class defined by them is somewhat stable and deserves further study.

Streaming and Sorting. The first extension is motivated by two facts about modern computing platforms. First, storage on disk is readily available and cheap (in the order of a few dollars per gigabyte). This implies that large amounts of temporary storage can, and should, be used, suggesting that the classical streaming model may be too restrictive in not allowing data storage at intermediate stages of the computation.

Second, sorting a large collection of fixed size records can be done extremely efficiently on commodity hardware, with rates that bottleneck the I/O subsystem (see for instance [Aga96, ADADC+97, Wyl99, Cha]). This contrasts with the fact that sorting is formally *hard* in streaming models. This seems to suggest that the formal streaming models are overly pessimistic in modeling the capabilities of today's computing platforms. If we assume that sorting is possible, i.e. we enhance the classical streaming model by providing a "sort box", then clearly the class of problems solvable by this "streaming and sorting model" is strictly larger than that solvable in the classical streaming model.

We formalize this model and show that it admits efficient solutions to a number of natural problems, such as undirected connectivity, minimum spanning trees, suffix array construction, and even some geometric problems. These problems are all known to be hard in the streaming model, suggesting that the addition of a sorting operation extends that model in a meaningful way.

To show the limits of this model, we state a function that cannot be computed efficiently in it, but allows for a simple computation by a machine that is allowed to read two streams in parallel.

Streaming Networks. Recent networking applications, such as the Chromium project [HEB+01, HHN+02] for distributed graphics rendering, use streams to exchange data in a network. The topology of such a network can be modeled as a directed acyclic graph. A node in the graph represents a machine with small local memory, and edges are streams transmitted between nodes. Each node reads the streams given by the edges pointing to it, and writes output streams for the edges leaving from it. We formalize this model, and address the question of its computational power.

It turns out that its capabilities depend heavily on the way in which the nodes can access their input streams. Forced to read them one after another in their entirety, the model becomes equivalent to the "streaming and sorting" model mentioned above. Allowed to freely interleave accesses to input streams, the

network's computational power increases dramatically. We show corresponding equivalences and separations.

During the course of the talk we will formalize the different models and present results of the form: A streaming algorithm with access to a "sort box", and using memory m is no more powerful than an external memory algorithm that uses $O(1)$ disks, requires main memory m and makes $O(\log n)$ "out-of-sequence" reads for each pass of the streaming algorithm.

More details can be found in the full version of this work [RADR].

References

[ADADC+97] Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, David E. Culler, Joseph M. Hellerstein, and David A. Patterson. High-performance sorting on networks of workstations. In *Proceedings SIGMOD*, pages 243–254, 1997.

[Aga96] Ramesh C. Agarwal. A super scalar sort algorithm for risc processors. In *Proceedings SIGMOD*, pages 240–246, 1996.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[Cha] Laurent Chavet. Fsort. Software. Available online at `http://www.fsort.com`.

[HEB+01] Greg Humphreys, Matthew Eldridge, Ian Buck, Gordon Stoll, Matthew Everett, and Pat Hanrahan. Wiregl: A scalable graphics system for clusters. In *Proceedings of SIGGRAPH*, 2001.

[HHN+02] Greg Humphreys, Mike Houston, Yi-Ren Ng, Randall Frank, Sean Ahern, Peter Kirchner, and James T. Klosowski. Chromium: A stream processing framework for interactive graphics on clusters. In *Proceedings of SIGGRAPH*, 2002.

[HRR99] Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. In *DIMACS series in Discrete Mathematics and Theoretical Computer Science*, volume 50, pages 107–118, 1999.

[MP80] J. Ian Munro and Michael S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12:315–323, 1980.

[RADR] Matthias Ruhl, Gagan Aggarwal, Mayur Datar, and Sridhar Rajagopalan. Streaming networks: Algorithms and complexity. Manuscript.

[Wyl99] Jim Wyllie. Spsort: How to sort a terabyte quickly. Technical report, IBM Almaden Research Center, 1999. Available online at `http://www.almaden.ibm.com/cs/gpfs-spsort.html`.