July 2008

MADALGO seminar by Jeremy T. Finemann, Massachusetts Institute of Technology

Cache-Oblivious Streaming B-Trees

The B-tree is the classic external-memory-dictionary data structure.
The B-tree is typically analyzed in a two-level memory model (called the DAM or I/O model) in which internal memory of size M is organized into size-B blocks, and there is an arbitrarily large external memory.

The cost in the model is the number of block transfers between internal and external memory. An N-element B-tree supports searches, insertions, and deletions in $O(\log_B N)$ block transfers.

In fact, there is a tradeoff between the cost of searching and inserting in external-memory dictionaries [Brodal, Fagerberg 03], and the B-tree achieves only one point on this tradeoff. A more general trade off is exhibited by their buffered B-tree.

This talk presents two points on the insert/search tradeoff for cache-oblivious (CO) data structures, the *cache-oblivious lookahead array (COLA)*, and the *shuttle tree*. The CO model is similar to the DAM model, except that the block size B and memory size M are unknown to the algorithm. The buffered B-tree is not cache oblivious---buffer sizes are chosen according to B.

The COLA implements searches in $O(\log_2 N)$ I/Os and inserts in amortized $O((\log_2 N) / B)$ I/Os. Notice that the searches are worse than in the B-tree by a $\log_2 B$ factor, but the inserts are better by a $B/\log_2 B$ factor. These bounds represent one optimal point on insert/search trade off space. In fact, when made into a cache-aware data structure, the *lookahead array* achieves the same trade off as the buffered B-tree.

The shuttle tree implements searches in the optimal $O(\log_B N)$ block transfers. Inserts cost $o(\log_B N)$ block transfers, improving on the B-tree by a superconstant factor.

Joint work with:
Michael A. Bender,
Martin Farach-Colton,
Yonatan R. Fogel,
Bradley C. Kuszmaul, and
Jelani Nelson