Cache-Oblivious and Cache-Aware Algorithms

Michael A. Bender*

A recent direction in algorithmic design and analysis is to pay particular attention to the structure of the memory hierarchy. Because memory hierarchies are growing “steeper,” the performance impact of the memory system is increasing dramatically. Many approaches and models have been developed for designing algorithms that interact well with the memory system, of which one of the more successful encompasses the body of work on *external-memory algorithms*, which effectively models two-level memory hierarchies (see, e.g., [AV88, Vit01]). Recently, the concept of *cache-oblivious algorithms* was proposed as a powerful theoretical approach with the potential for substantial practical impact [FLPR99, Pro99]. The idea is to hide any parameters of the memory hierarchy—such as block transfer sizes and the size of each memory level—from the algorithm. This simple idea has powerful ramifications:

1. Cache-oblivious algorithms automatically adapt to arbitrary memory hierarchies.
2. Cache-oblivious algorithms can be analyzed on a simple two-level memory hierarchy, and then automatically perform as well on a complex multilevel memory hierarchy with particular page replacement strategies, limited associativity, etc.

Motivated by these exciting consequences, an increasing number of researchers have started to develop algorithms and data structures in this model. In the past 3 years, there have been over fifteen papers developing efficient cache-oblivious algorithms and data structures. The field has been shown to have substantial depth and has led to new general techniques for maintaining data locality in a memory hierarchy. Yet the field of cache-oblivious algorithms is still in its infancy. Practical and theoretical issues remain unsolved and ripe for exploration, bridging the gap between the best algorithms known in the external-memory and cache-oblivious contexts.

This talk gives a overview of cache-oblivious techniques and algorithms. Particular focus will be given to cache-oblivious matrix multiplication [FLPR99, Pro99], sorting [FLPR99, Pro99, BF02a], and search-trees [BD1W02, BFK02, BDFC00, RCR01, BCR02], as well to to some of the open questions in the area.

---

*Department of Computer Science, SUNY Stony Brook, Stony Brook, NY 11794-4400, USA; bender@cs.sunysb.edu.
References


