External-Memory Algorithms

Jeff Vitter
Duke University

Abstract

In many applications, data sets are too massive to fit completely inside a computer's internal memory. The input/output communication (or I/O) between the fast internal memory and the slower external memory (such as disks) can be a performance bottleneck. In this tutorial, we survey the state of the art in the design and analysis of external-memory algorithms. We seek to optimize two measures of performance: number of I/O operations and CPU time. Our model is general enough to handle multiple disks (or disk arrays) and parallel CPUs; related scenarios include cache hierarchies, hierarchical memory, and tertiary storage.

The canonical external-memory problem is sorting. We explain the fundamental lower bounds on the number of I/Os needed to sort and permute in external memory. We show how to sort optimally by use of the distribution and merging paradigms. External sorting arises indirectly in the solution of some other problems, in which efficient external-memory algorithms can be obtained by simulation of well-known parallel algorithms. Other useful techniques for batched problems that we survey include distribution sweep (useful for spatial join and all nearest neighbors), buffer trees (for priority queues, sorting, and batched R-tree construction), and batched filtering and external fractional cascading (for GIS map overlay and red-blue line segment intersection). In the online setting, in which the data can be updated and queried, we discuss several important external memory data structures, such as B-trees (for dictionaries and one-dimensional range queries), interval trees with weight-balanced B-trees (for stabbing queries), and R-trees, quad trees, and other hierarchical storage schemes (for applications such as multidimensional range queries and contour line processing in GIS).