LA SAPIENZA

Cache-oblivious algorithms

Irene Finocchi

Dept. of Computer and Science

Sapienza University of Rome

Towards a theoretical model

- Improve temporal and spatial locality
- Take care of data access patterns and block data
- Parameter tuning (cache size, block size, associativity, page replacement, etc)

How can we formalize these ideas and get a theoretical model in which we are able to analyze cache-efficiency?

Open question for quite a while

Ideas in the air, not a systematic theory

Cache-oblivious model (1999)

Cache-oblivious model

Idea:

- design cache-friendly algorithms without knowing cache parameters (internal details of the memory hierarchy)
- analyze algorithms using cache parameters

Simple idea, with several surprisingly powerful consequences.

Introduced by Frigo, Leiserson, Prokop & Ramachandran in FOCS'99

Implications of cache obliviousness

• If cache-oblivious algs perform well between two levels of the memory hierarchy, then must automatically work well between any two adjacent levels of the hierarchy.

⇒ we design algorithms in a two-level model, and algorithms automagically adapt to the whole hierarchy

• Self-tuning: cache-oblivious algs work well on all machines without modification (still subject to some tuning, e.g., where to trim base case of recursion) ⇒ code portability

• In contrast to external-memory model, cache-oblivious algs cannot explicitly manage the cache

Ideal cache model

How can we design algs that minimize number of block transfers if we do not know the page-replacement strategy?

An adversarial page replacement strategy could always evict next block that will be accessed…

Cache-oblivious model assumes an ideal cache:

- page replacement is optimal
- cache is fully associative

Assumption 1

Optimal Page Replacement:
Page replacement strategy knows the future and always evicts page that will be accessed farthest in future.

Real-world caches do not know the future, and employ more realistic page replacement strategies such as evicting the least-recently-used block (LRU) or evicting the oldest block (FIFO).

Assumption 2

Full Associativity

Any block can be stored anywhere in cache (all cache lines in the same set, S=1, E=C/B)

Most caches have limited associativity: memory addresses can be mapped to a small subset of cache lines (i.e., to lines in the same set).

Typical real-world caches: 8-way, 16-way, even less (depends on platform)

Justification of ideal cache model

Frigo *et al*. justify the ideal-cache model by a collection of reductions that modify an ideal-cache alg to operate on a more realistic cache model.

Running time of the alg. degrades somewhat, but in most cases by only a constant factor.

(Hey, wait a minute! We are doing algorithm engineering: we're interested in constants!)

Won't go into the details of the proofs.

Justification of assumption 1

First reduction: *replacement strategy*
Remove optimal (omniscient) replacement strategy that uses information about future requests.

If an alg makes T memory transfers on cache of size M/2 with optimal replacement, then it makes at most 2T memory transfers on cache of size M with LRU or FIFO replacement (and same block size B).

I.e., LRU and FIFO do just as well as optimal replacement up to constant factors of memory transfers and cache size. This competitiveness property of LRU and FIFO goes back to a 1985 paper of Sleator and Tarjan.

Justification of assumption 2

Second reduction: *associativity and automatic page replacement*

Convert full associativity into direct-mapped cache and automatic replacement into manual memory management

Another assumption: tall cache

Recall $C = S \times E \times B$, here $S=1$

Commonly assumed that cache is taller than wide, i.e., number of cache lines, $E=C/B$, larger than size B of each line:

$$C = \Omega(B^2)$$

Already seen in external-memory algorithms.
Usually true in practice