Optimal Web Cache Sizing: Scalable Methods for Exact Solutions

Terence Kelly & Daniel Reeves

{tpkelly,dreeves}@eecs.umich.edu

Electrical Engineering & Computer Science
University of Michigan
Ann Arbor MI 48109-2110 USA

23 May 2000

$Id: wcpsl.tex,v 1.3 2000/05/16 19:23:45 t-tkelly Exp $

Outline

- Motivation
- Starting point: The five-minute rule
- Branching 2-level hierarchy, idealized workload & cost models
- Single cache, more realistic workload & cost models
- Fast simultaneous simulation
  - for cache sizing
  - for workload characterization
- Summary

Economically Optimal Cache Sizes

The boss asks, "Why aren't our caches larger or smaller?" *Reply with a business argument.* ("Because the company would lose money if they were larger or smaller.")

Monetary tradeoffs of Web caching important in large-scale systems, e.g., WebTV. Also important to industry analysts:

> [Cache Vendor X] is targeting the enterprise, where most network managers will be loath to spend $40,000 to save bandwidth on a $1,200-per-month T1 line. To sell these boxes, [Vendor X] must wise up and deliver an entry-level appliance starting at $7,000.
>
> —*Forrester Research Tech Report, 01/98*

Challenge: Decentralized control (no end-to-end ownership).

The Back of the Envelope

requests $\longrightarrow$ Cache $\$_M$ $\overset{\$_B}{\text{———}}$ Origin Servers

Given a document of size S bytes that is reqested f times per second, we can

- cache it at a cost of $\$_M$ dollars per byte, or
- repeatedly retreive it across a link on which bandwidth costs $\$_B$ dollars per (byte/sec)

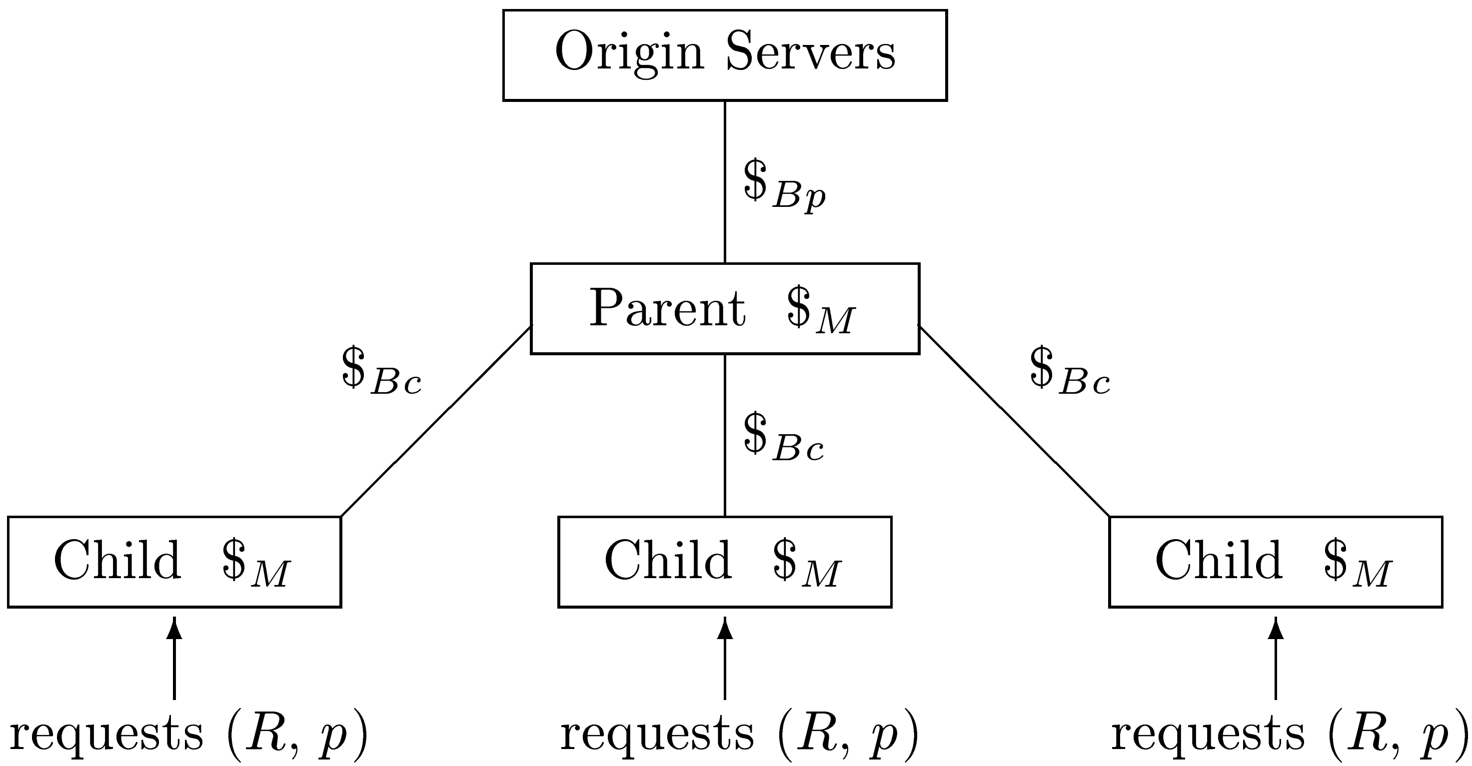
Cache document when storage is cheaper than bandwidth:

$$\$_M S \;\leq\; \$_B f S \quad \Rightarrow \quad f \geq \frac{\$_M}{\$_B}$$

Optimal cache size: just big enough to hold all documents that satisfy above condition.

Gray & Putzolu, "The 5 minute rule...", *Proceedings of ACM SIGMOD*, May 1987.

Two-level Cache Hierarchy



Model Assumptions

- *Identical* reference streams at child caches
- Independent reference model
- All caches use Perfect-LFU replacement

Determine optimal cache sizes by deciding where to cache individual documents (at parent, children, or neither) and sizing caches to accommodate them.

Centralized & Decentralized Optimization

Centralized: An omniscient “central planner” computes globally-optimal cache sizes for all caches based on all costs and workloads.

Decentralized: Each cache computes a size for itself that minimizes local expenditures based on local workload and costs.

Result: If the number of child caches is large, centralized & decentralized computations yield the same cache sizes.

Scalable decentralized computation entails no penalty in solution quality.

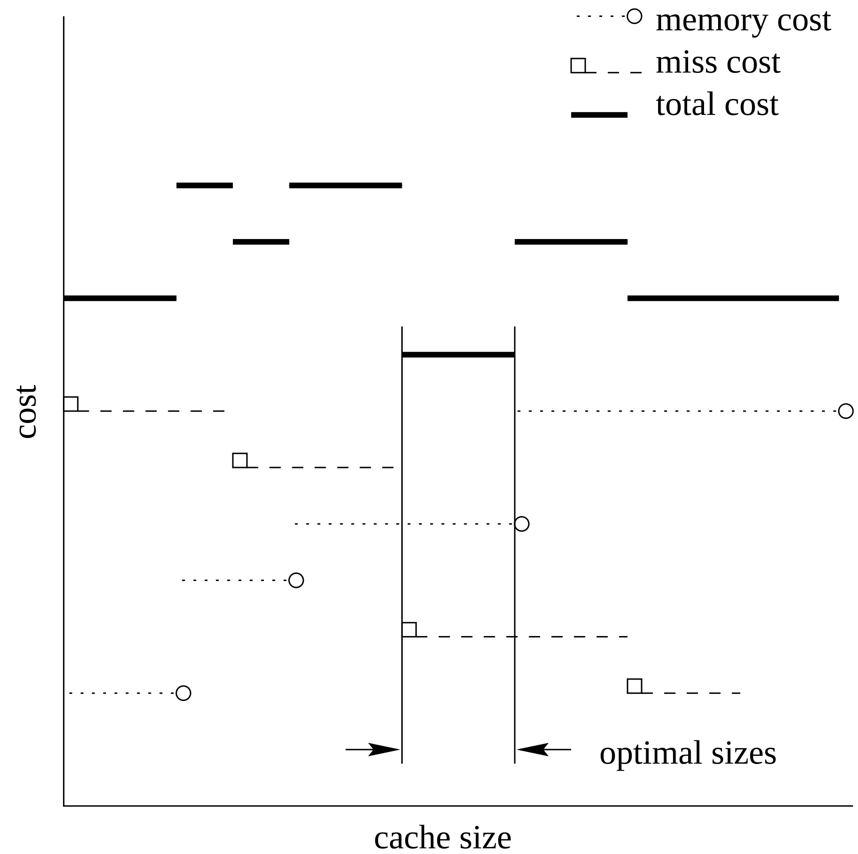
Single Cache, Realistic Assumptions

Assumptions so far (independent reference model, Perfect-LFU replacement, linear cost functions) are not very realistic.

Consider optimizing size of a *single cache* under more plausible assumptions:

- Workload described by an explicit sequence of requests
- Arbitrary miss cost associated with each request
- Cost of memory is an arbitrary monotonic function
- Cache uses LRU replacement

Cost functions



Computing aggregate miss cost as a f(cache size) is the hard part.

Fast Simultaneous Simulation Algorithm

We describe an algorithm that

- Computes aggregate miss cost as a function of cache size
- Makes *one* pass over reference sequence, i.e., simultaneously computes miss cost at *every* cache size of interest
- Requires $O(M \log M)$ time and $O(M)$ memory, where M is the number of references in the input sequence
- Special case: Computes LRU stack distance transformation of input sequence (a measure of temporal locality) in $O(M \log N)$ time and $O(N)$ memory, where N is the number of unique documents referenced

Very Fast, Surprisingly Unoriginal

Martin Arlitt's tests on very large traces (1.35 billion references) show that our algorithm computes stack distances much faster than alternatives on inputs with low-to-moderate temporal locality.

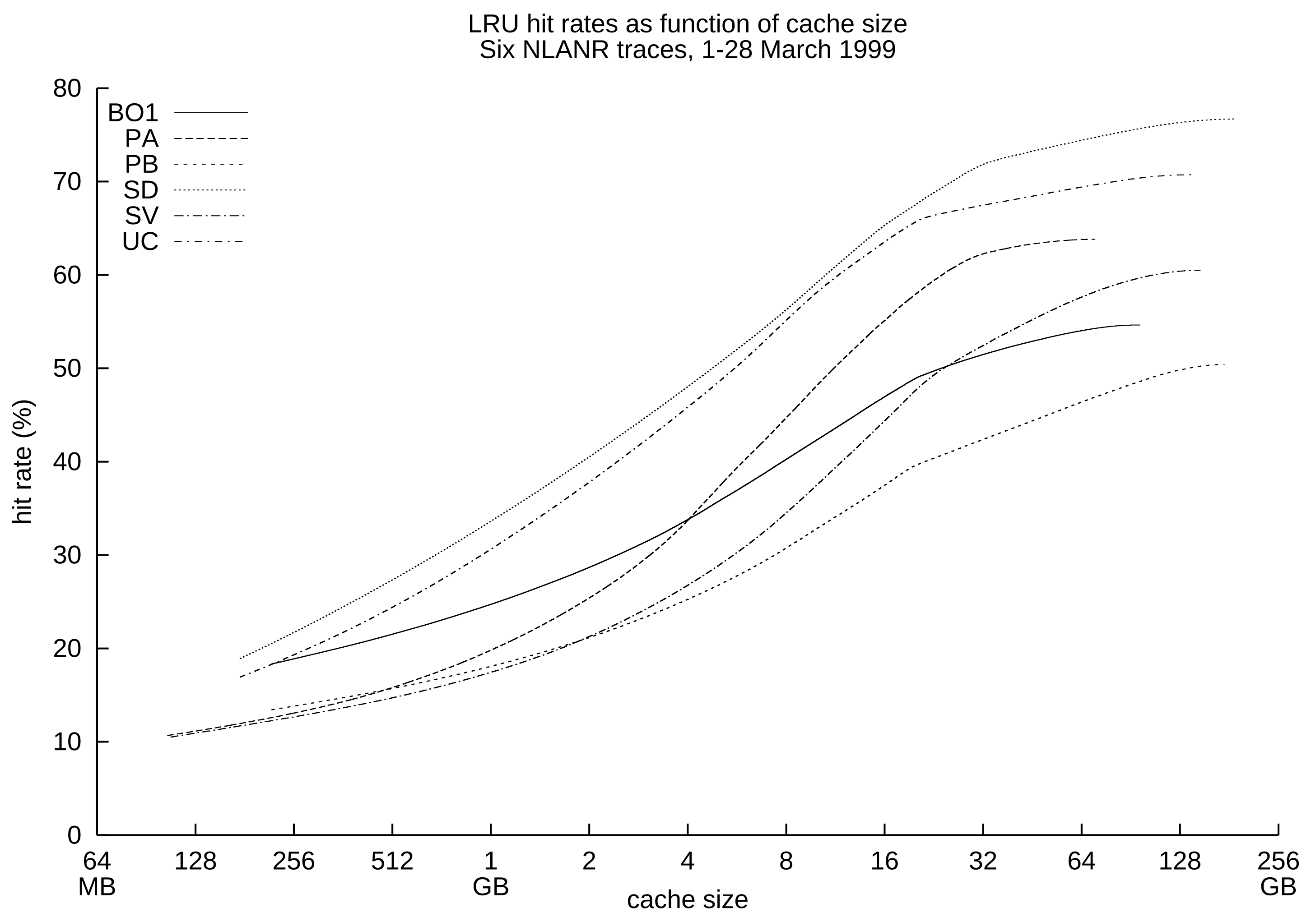
Very similar algorithms are described in mid-1970s and 1980s processor-memory literature, but not widely cited in modern Web caching literature.

Several researchers use $O(MN)$-time stack distance algorithms today, and were surprised to learn that far faster methods exist.

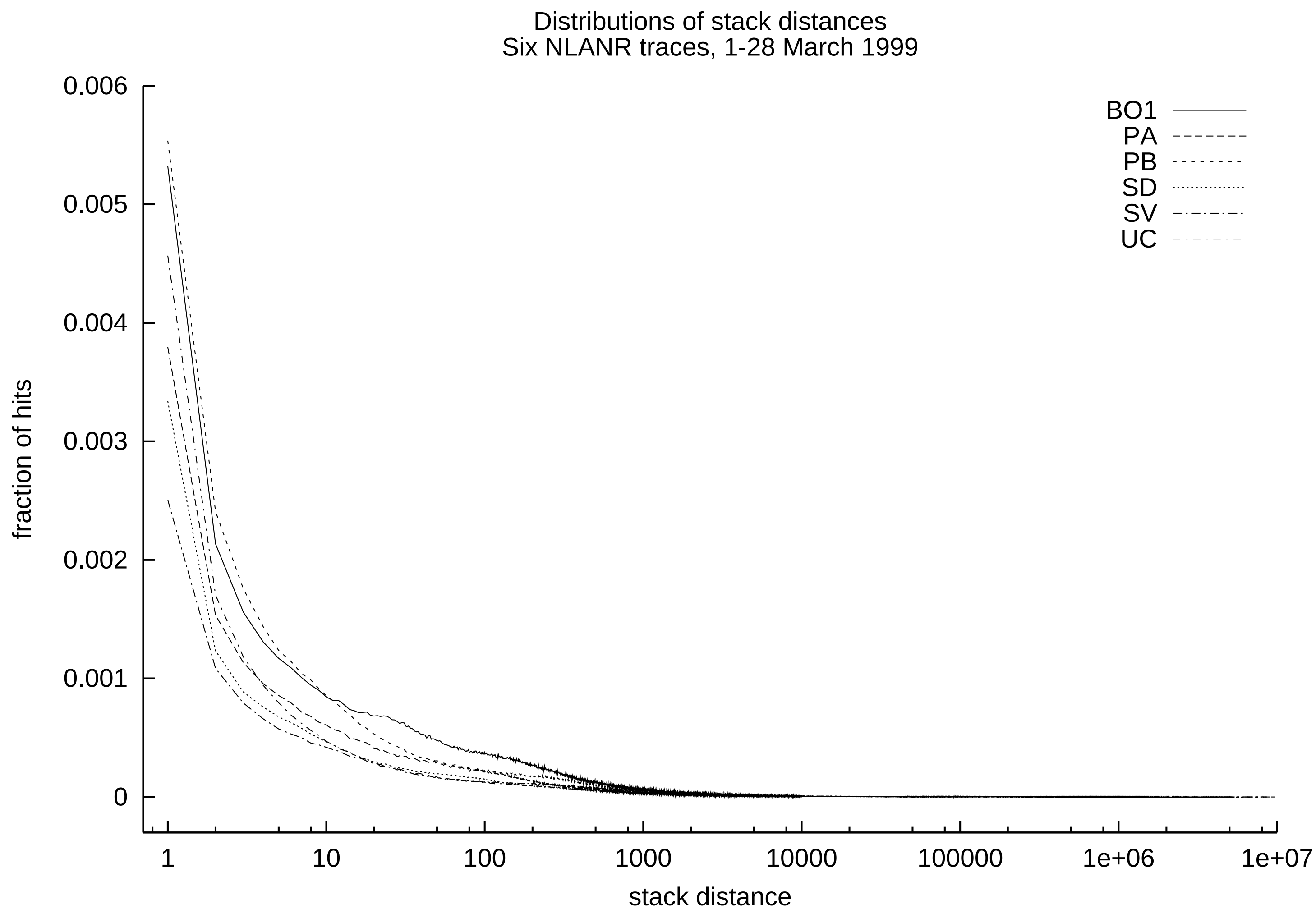
ANSI C implementation of our algorithm is available.

Mattson et al. 1970; Bennett & Kruskal 1975; Olken 1981; Thompson 1987.

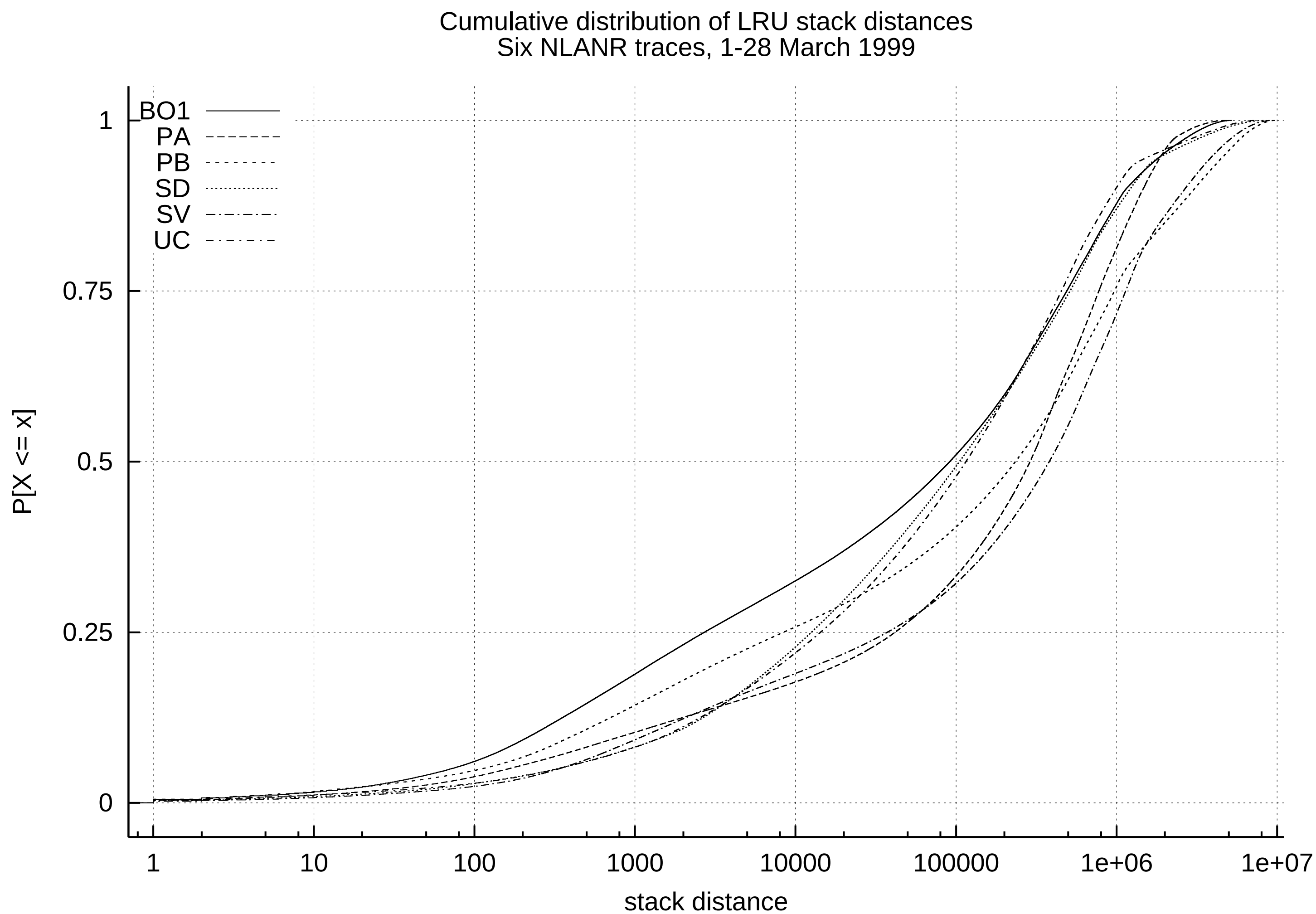
Hit rates as f(cachesize)



Distribution of Stack Distances



Cumulative Distribution of Stack Distances



Summary

- Straightforward to generalize five-minute rule to branching cache hierarchies
- Decentralized computation yields globally-optimal cache sizes in the limit as system size grows large—scalability where it's needed most
- We describe a surprisingly unoriginal, very fast algorithm for computing arbitrarily-weighted hit rates & stack distances
- Fast algorithm optimizes single cache size under plausible assumptions
- Future work: account for consistency issues & document expirations