A Study of Integrated Prefetching and Caching Strategies

Pei Cao, Edward W. Felten, Anna R. Karlin and Kai Li

ACM SIGMETRICS Performance Evaluation Review, 1995

presented by Scott Schneider
Motivating Example

- Consider the reference stream “ABCA” where
  - The file cache holds two blocks
  - Fetching a block takes four time units
  - “A” and “B” are initially in the cache
Motivating Example

- No-prefetch policy for “ABCA”
Motivating Example

- No-prefetch policy for “ABCA”
Motivating Example

- No-prefetch policy for “ABCA”
Motivating Example

- No-prefetch policy for “ABCA”
Motivating Example

- No-prefetch policy for “ABCA”
Motivating Example

- No-prefetch policy for “ABCA”
- 8 time units
Motivating Example

- Always prefetch policy for “ABCA”
Motivating Example

- Always prefetch policy for “ABCA”
Motivating Example

- Always prefetch policy for “ABCA”
Motivating Example

- Always prefetch policy for “ABCA”
Motivating Example

- Always prefetch policy for “ABCA”
Motivating Example

- Always prefetch policy for “ABCA”
- 10 time units
What Happened?

- Aggressive prefetching is not always beneficial
  - The aggressive approach resulted in an unnecessary prefetch operation
  - Prefetching can hide latency, but can result in more total fetches

- The problem: given an access pattern, what is the optimal combined prefetching and caching strategy?
Outline

- Problem Motivation
- Previous Work
- Caching Background
- Theoretical Analysis
- Simulation Results
- Conclusions
- Impact of the Work
Previous Work

- Three classes: exploiting locality on disk, general prefetching, and prediction
- Locality on disk
  - One-block lookahead first proposed by Alan Jay Smith in 1978; assumes if $k$ and $k + 1$ blocks were referenced, $k + 2$ block will be referenced
  - Clustering disk operations, McVoy and Kleiman, 1991
Previous Work

- **Prefetching**
  - Using application hints to inform prefetching; Patterson, Gibson, and Satyanarayanan, 1993
  - Prefetch based on past accesses using probability graphs, Griffioen and Appleton, 1994

- **Prediction**
  - Detect user file access patterns at runtime, Tait and Duchamp, 1990
  - Train an associative memory to recognize access patterns, Palmer and Zdonik, 1991
  - Data compression techniques, Vitter and Krishnan, 1993
Caching Background

- Define our model
- Present the Four Rules
- Introduce two prefetching policies that obey the Four Rules
Model Definition

- Known sequence of accesses \((r_1, r_2, \ldots, r_n)\), which means this is offline
- Cache holds \(k\) blocks, \(k < n\)
- Cache hits are satisfied immediately, otherwise the block is fetched from disk
Model Definition

- **Fetches**
  - Can result from a miss (on-demand) or in anticipation of a miss (prefetch)
  - Take $F$ time units, stall time can be $F - i$ if it was started at time $i$
  - Happen one at a time
  - Invalidate one item in the cache at the start of the operation
Model Definition

- Total execution time is the total reference time plus the total stall time
- Our goal is to minimize the total execution time by making the following decisions
  - When to fetch a block
  - Which block to fetch
  - Which block to replace
The Four Rules

- Any optimal prefetching strategy must follow these rules
  - **Rule 1: Optimal Prefetching** Every prefetch should bring in the next block in the stream that is not in the cache
  - **Rule 2: Optimal Replacement** Every prefetch should discard the block whose next reference is farthest in the stream
  - **Rule 3: Do No Harm** Never discard a block when its replacement is referenced later in the stream
  - **Rule 4: First Opportunity** Always perform a prefetch-replace operation as soon as it is possible
Two Prefetching Policies

- Two simple policies that obey all four rules are *conservative* and *aggressive*
  - *Conservative* performs the minimum number of prefetches
  - *Aggressive* prefetches the next missing block at the earliest opportunity, replacing the block that is farthest in the stream
Theoretical Results

- Looking at *conservative* and *aggressive*
  - *Conservative* performs the same number of fetches as the optimal off-line demand paging algorithm
  - *Aggressive* performs at most the number of fetches as LRU demand paging
  - If $F < k$, the elapsed time of aggressive is at most $(1 + F/k)$ times optimal; if $F > k$, elapsed time is at most twice optimal
  - The tight bound of aggressive is $1 + (F - 2)/(k + 1)$ times optimal
  - Elapsed time of *conservative* is at most twice optimal
  - Tight bound of conservative is $2(1 - 1/F)$ times optimal
Bounds on Fetches

- *Conservative* performs the same number of fetches as optimal, but elapsed time is often smaller
  - This is due to overlapping fetches with references
  - Unable to do this rarely, such as cyclic references
Bounds on Fetches

- **Aggressive** performs at most the same number of fetches as LRU
  - On patterns with strong locality, *aggressive* may have near optimal performance because it will not have excessive fetches and there will be significant overlap
Simulation: Other Approaches

- Six other algorithms used in simulation
  - *LRU-demand* fetches a page on demand, and replaces the LRU block
  - *OPT-demand* fetches a page on demand, and replaces the block furthest in the reference stream
  - *LRU-OBL* is the same as *LRU-demand* but with one-block lookahead
  - *OPT-OBL* is the same as *OPT-demand* but with one-block lookahead
  - *LRU-sensible* fetches the next missing block at earliest opportunity, replaces the LRU block
  - *LRU-throttled* uses 1/3 of the cache for prefetches
Simulation: Traces

- Running read-dominated applications on a workstation
  - cscope searches for strings in large software packages
  - dinero is a cache simulator
  - glimpse is a text information retrieval system
  - postgres-join is a relation database; performs a join

- File server
  - Five sets of traces from 40 clients' file activities
  - Gathered over 48 or 24 hours
  - Not read dominated
Simulation: Models

- **Simplified**
  - Same as previously presented theoretical model
  - “Reference time” is uniform and defined as one time unit
  - Fetches take $F$ time units

- **Realistic**
  - Uses actual time interval between accesses
  - Uses measured average fetch time
  - Only applies to workstation traces
Simulation: Model Validity

- Even the *realistic* model ignores many characteristics of real disks
- However, simulated elapsed time and number of fetches is within 10-15% of measured
- Tuned *realistic* to have same parameters of *simplified*, differences were within 2%
Simulation: Results

- File server traces, *simplified* model
Simulation: Results

- Workstation traces, realistic model

![Graph showing simulation results](image_url)
Simulation: Summary

- *Aggressive* performs best, *conservative* slightly worse, *LRU-sensible* best of the LRU based algorithms
- As the cost of disk access increases, replacement policies dominate over prefetching policies
Impact

- Citations peaked in 1998 with 20; 122 citations total, including into 2004
- Other authors cite this paper for
  - The four rules for optimal prefetching and caching
  - Studying the interaction of cache and prefetching strategies and when cache dominates
  - Overlapping CPU and I/O
Follow-up

- Improvements
  - Addresses multiprocessor case
  - Includes disk scheduling with caching and prefetching
- Online, but requires external knowledge in the form of application hints
- Implementation and Performance of Integrated Application-Controlled Caching, Prefetching and Disk Scheduling
  Pei Cao, Edward W. Felton, Anna R. Karlin and Kai Li
  ACM Transactions on Computer Systems, 1996