R. Hugo Patterson et al.: Informed Prefetching and Caching (SOSP 1995)

Björn Döbel

31.01.2007

Contents

Motivation (1)

Intention

Overcome *reactive* buffer cache management by using *proactive* strategies.

Reasons

1. Increasing level of disk parallelism
2. File access performance grows more important
3. I/O-intensive applications can give hints about their demands

Motivation (2)

Scenarios

- read-intensive applications in general
- text search
- scientific visualization
- database queries
- pattern recognition
- object linkers

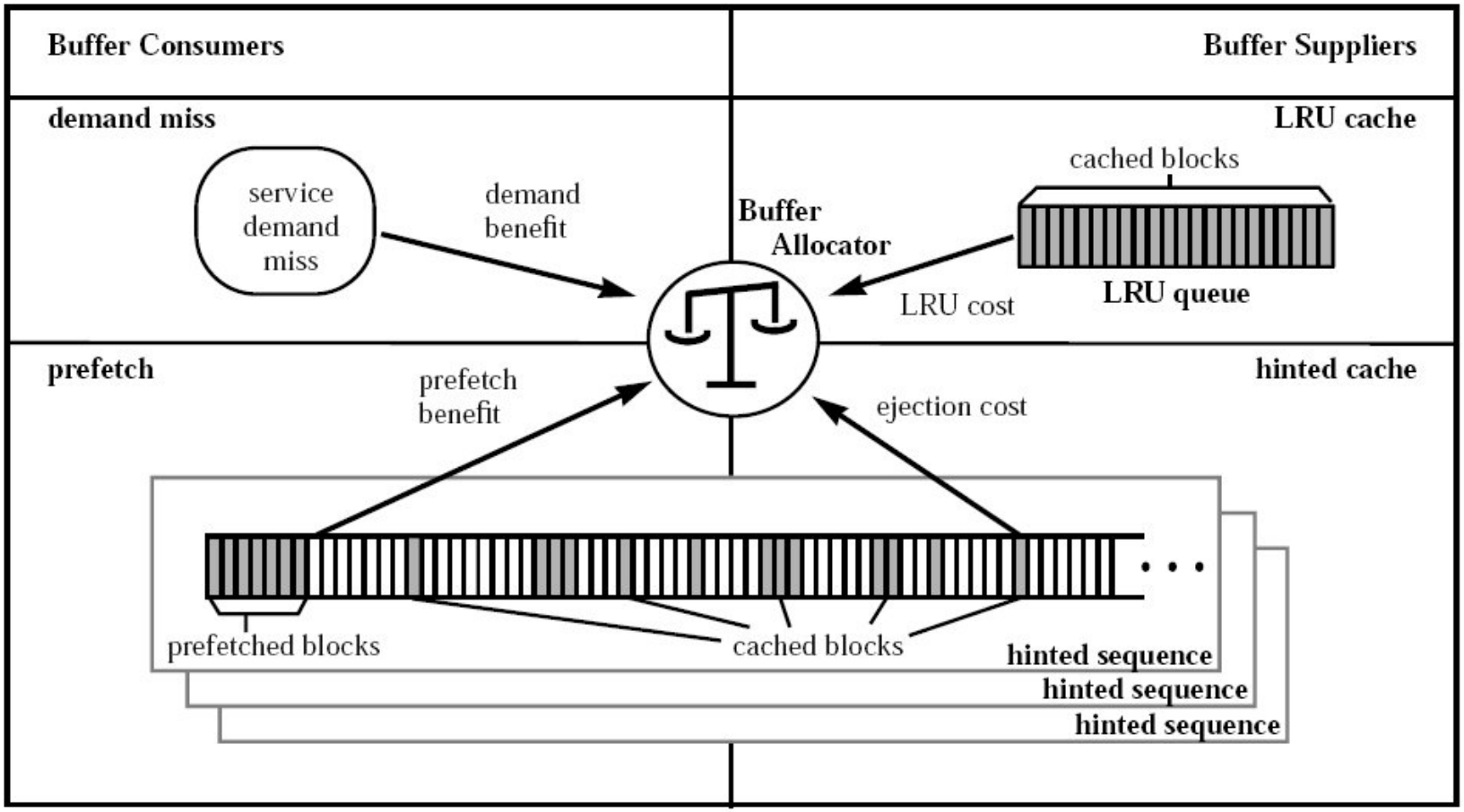
Problems

- prefetching may conflict with caching of buffers
- analyse cost of cache ejection vs. benefit of prefetching to decide when to prefetch

Hints

- *Disclosing hints*: use existing interface to give hints — exploits knowledge about application behavior
- *Advising hints*: give hints about cache policy, number of accesses, . . . — exploits knowledge about application and the underlying system
- authors prefer disclosure over advice because:
  - it is system-independent
  - it is more robust, because the OS may use it along with its own policy
  - it is a modular approach

System model



Terms

- *Estimator* - function to estimate the overall cost of a buffer in cache or to be prefetched
- *Common currency:* magnitude of change in I/O service time per buffer access
- *Time for a cache miss:* $T_{miss} = T_{hit} + T_{driver} + T_{disk}$

Benefit of allocating a buffer for a consumer

Allocating a buffer for prefetching leads to a stall time $T_{stall} \leq T_{disk}$:

$$T_{pf}(x) = T_{hit} + T_{driver} + T_{stall}(x)$$

The benefit of using one more page for prefetching is:

$$\begin{aligned} \Delta T_{pf}(x) &= T_{pf}(x+1) - T_{pf}(x) \\ &= T_{stall}(x+1) - T_{stall}(x) \end{aligned}$$

Evaluating stall time

- Assumption: x-th disk block is needed no sooner as $x(T_{CPU} + T_{hit} + T_{driver})$.
- Fetching block will cost T_{disk}.
- Then $T_{stall}(x) \leq T_{disk} - x(T_{CPU} + T_{hit} + T_{driver})$
- Benefit of prefetching is decrease of stall time.
- If x-th block is needed further in the future than T_{disk}, T_{stall} increases no more $\rightarrow$ *prefetch horizon* $P(T_{CPU})$:

$$P(T_{CPU}) = \frac{T_{disk}}{T_{CPU} + T_{hit} + T_{driver}}$$

Benefit of allocating a buffer for a consumer (2)

- In fact, stall times overlap, so that we need only to account one stall time for every x prefetch operations:

$$T_{stall} = \frac{T_{disk} - x(T_{CPU} + T_{hit} + T_{driver})}{x}$$

- This leads to a function for estimating prefetch benefit:

$$\Delta T_{pf}(x) = \begin{cases} x = 0 & -(T_{CPU} + T_{hit} + T_{driver}) \\ x < P(T_{CPU}) & \frac{-T_{disk}}{x(x+1)} \\ x \geq P(T_{CPU}) & 0 \end{cases}$$

Cost of shrinking the LRU cache

- Given a certain cache hit ratio $H(n)$ for n cache pages, we can determine the decrease in hit ratio when removing a page from LRU cache:

$$\begin{aligned} T_{LRU}(n) &= H(n)T_{hit} + (1 - H(n))T_{miss} \\ \Delta T_{LRU}(n) &= T_{LRU}(n-1) - T_{LRU}(n) \\ &= (H(n) - H(n-1))(T_{miss} - T_{hit}) \end{aligned}$$

Cost of ejecting a hinted block

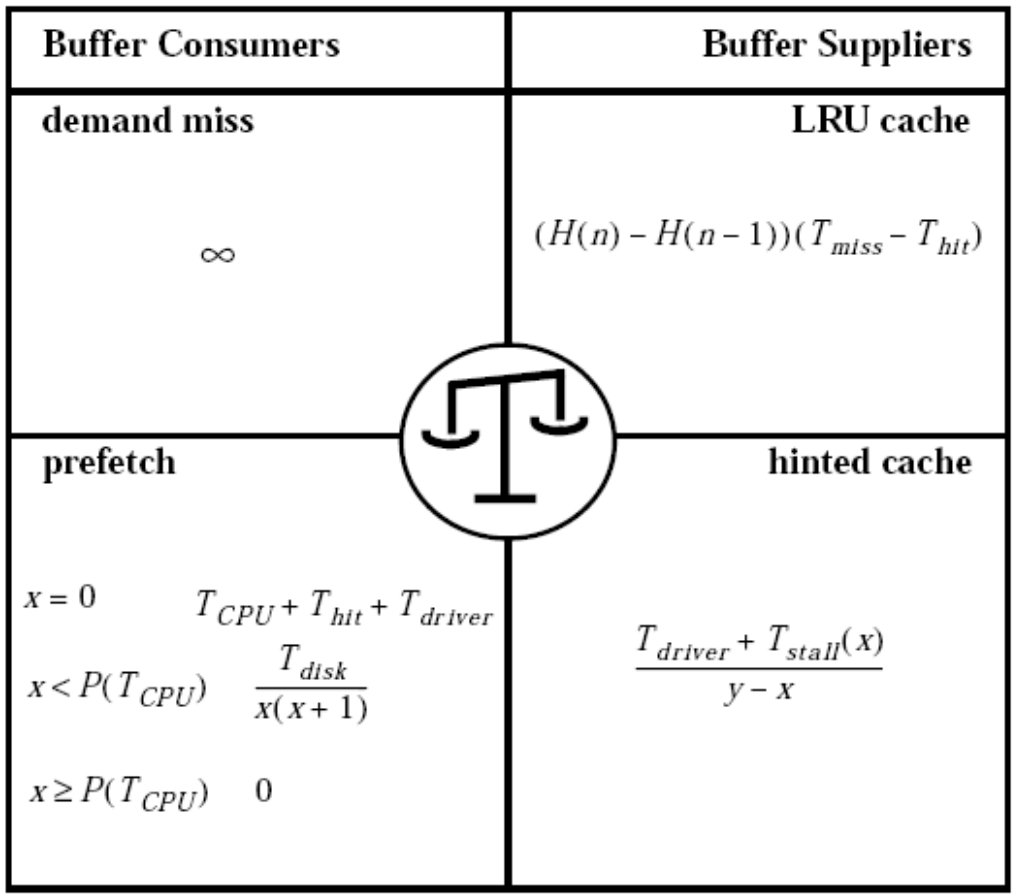
- Ejecting a hinted block increases T_{hit} up to T_{pf}, therefore we have:

$$\begin{aligned} \Delta T_{eject}(x) &= T_{pf}(x) - T_{hit} \\ &= T_{driver} + T_{stall}(x) \end{aligned}$$

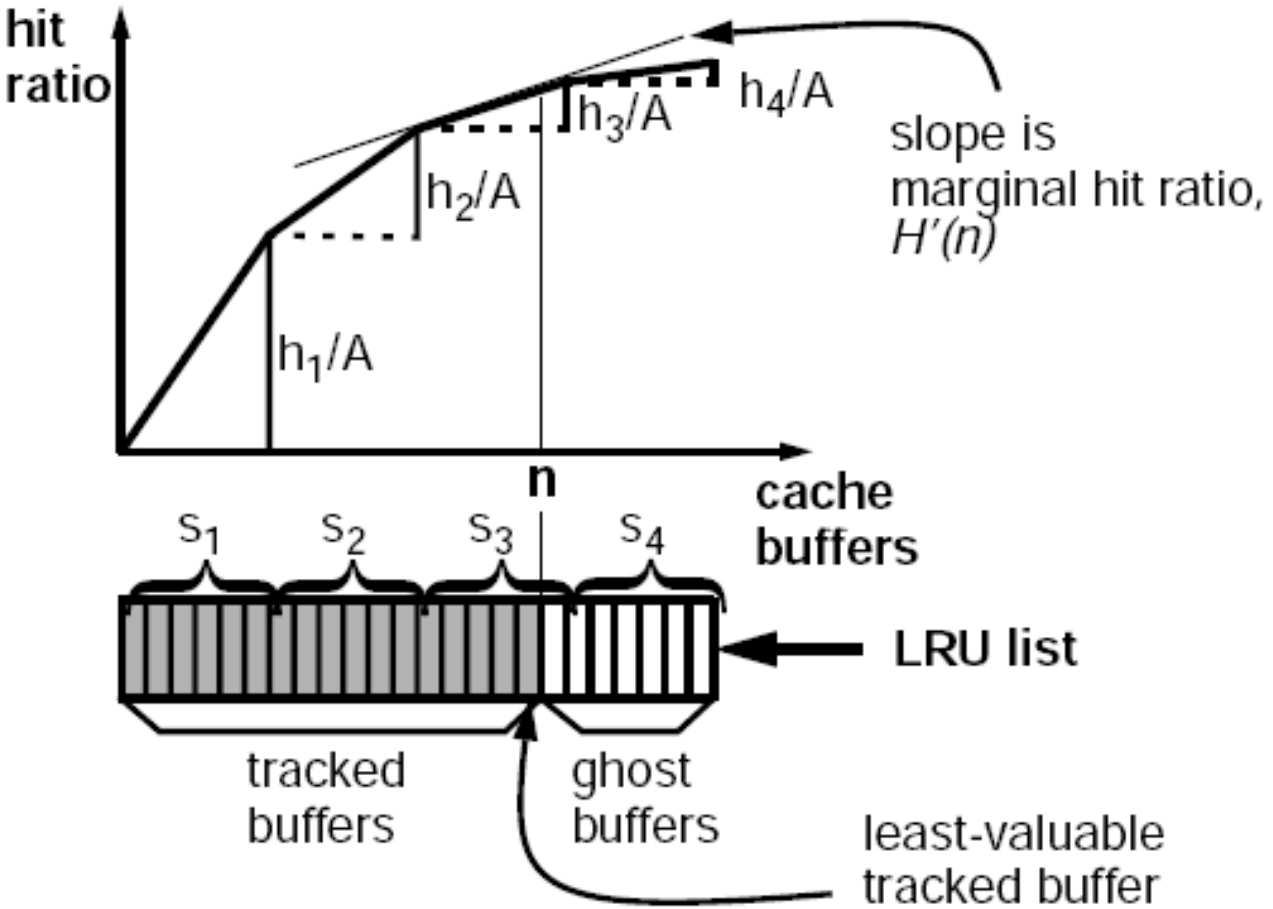
- However, when ejecting a block that is needed in y time, we need to re-prefetch in x and therefore have $y - x$ to accomodate the cost of ejecting. Therefore:

$$\Delta T_{eject}(x, y) = \frac{T_{driver} + T_{stall}(x)}{y - x}$$

Putting all together

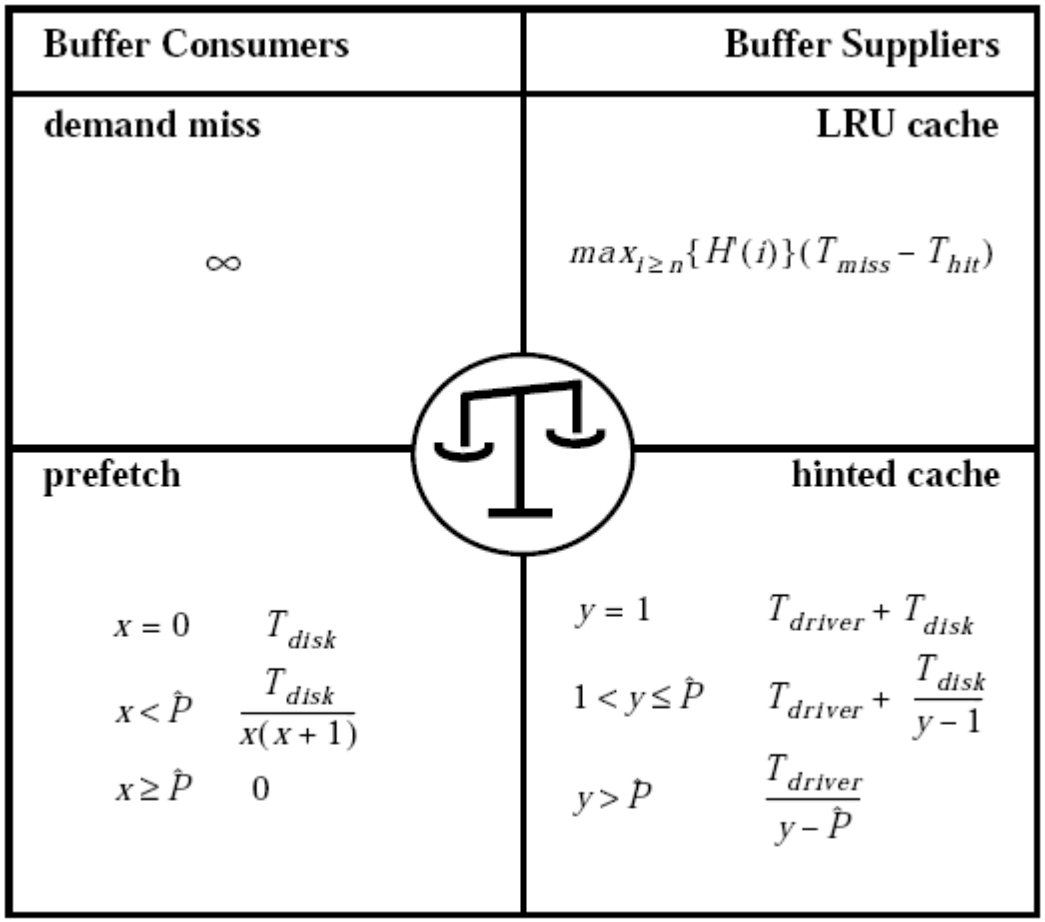


Determining $H(n)$



$$\Delta H(n) \approx H'(n) \approx \frac{h_i}{A|s_i|}$$

Implementation



Results

- Single-application workloads become faster using hints
- Increasing number of disks in array leads to decrease in I/O time
- Multip-programmed workloads may experience problems

Further Reading

- Chang, Gibson: *"Automatic I/O Hint Generation through Speculative Execution´´*, OSDI 1999
- Desai, Huizinga: *"Implementation of Informed Prefetching and Caching in Linux´´*, ITCC 2000

- From today's point of view, have we now reached the point, where RAM is cheap enough, so that we don't need prefetching anymore?
- Linus Torvalds on LKML in 2002:

  The device [..] does a lot better at readahead than higher layers can do anyway.