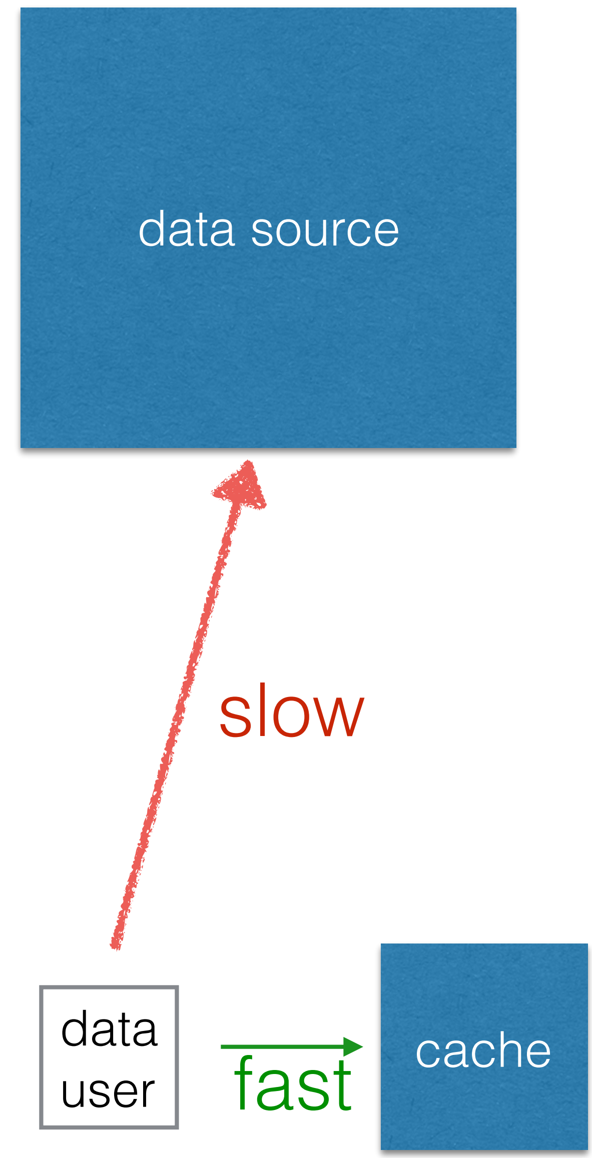
Operating Systems

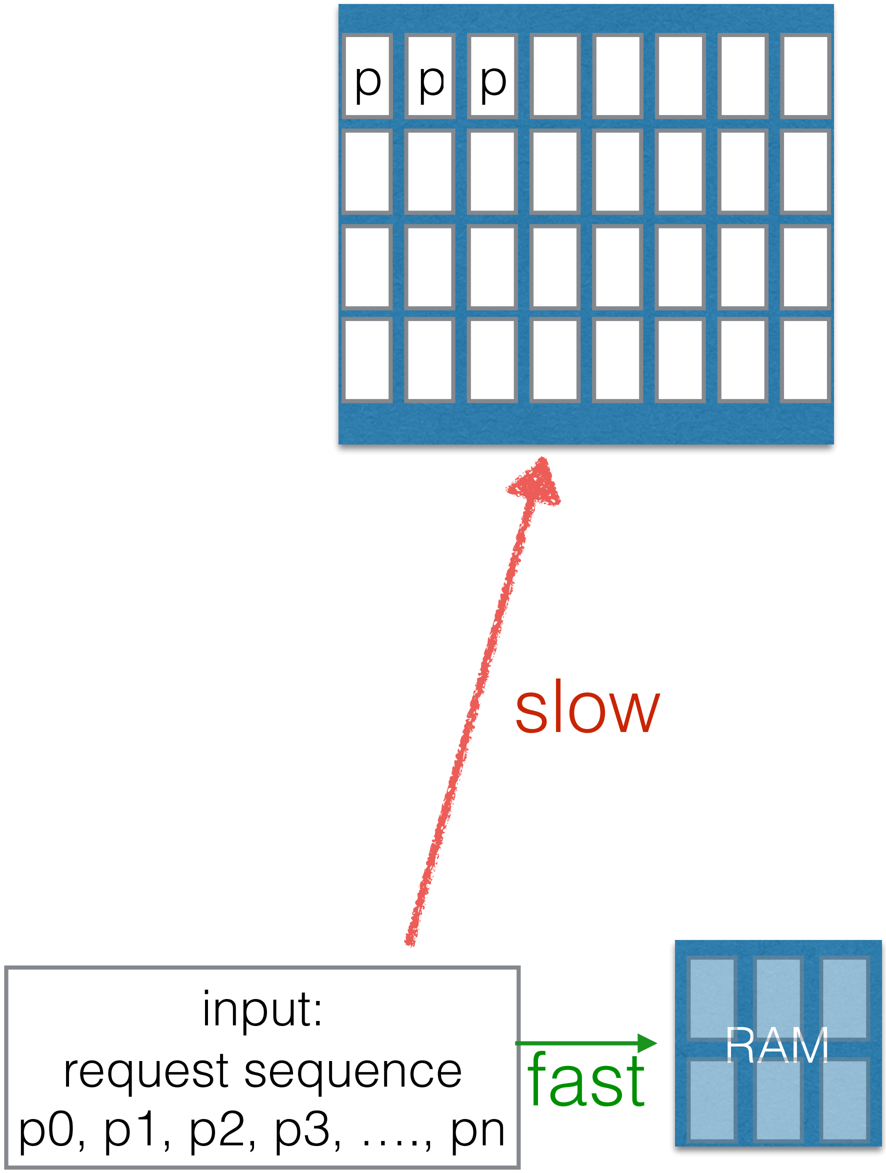
Memory: replacement policies

Last time

- **caching speeds up data lookups if the data is likely to be re-requested again**
- **data structures for O(1)-lookup**
  - set-associative (hardware)
  - hash tables (software)
- **cache coherence**
  - with 1 cache: buffer writes back to memory
  - with multiple caches: things get complicated
- today: **replacement policy**



Model of virtual memory

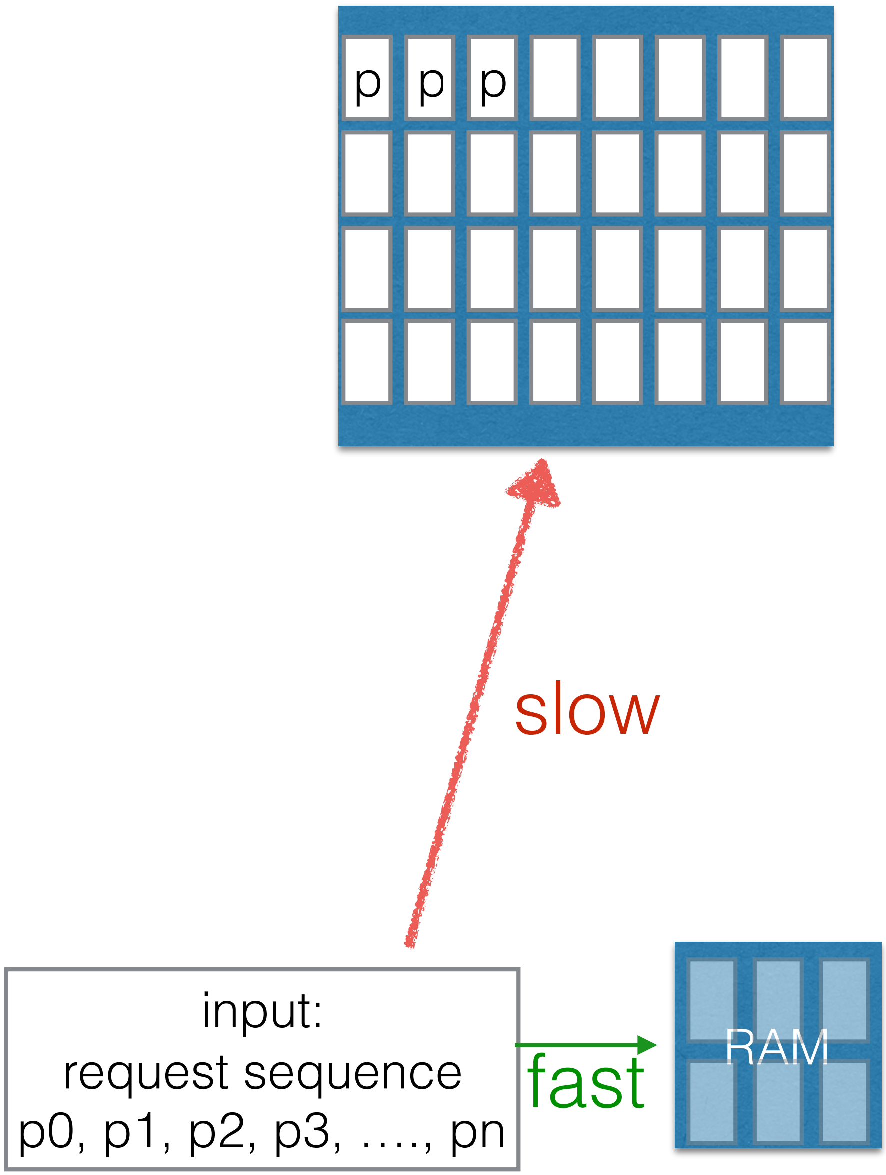


disk: slow memory

cache of size **k**

k much smaller than # of pages in disk

Model of virtual memory

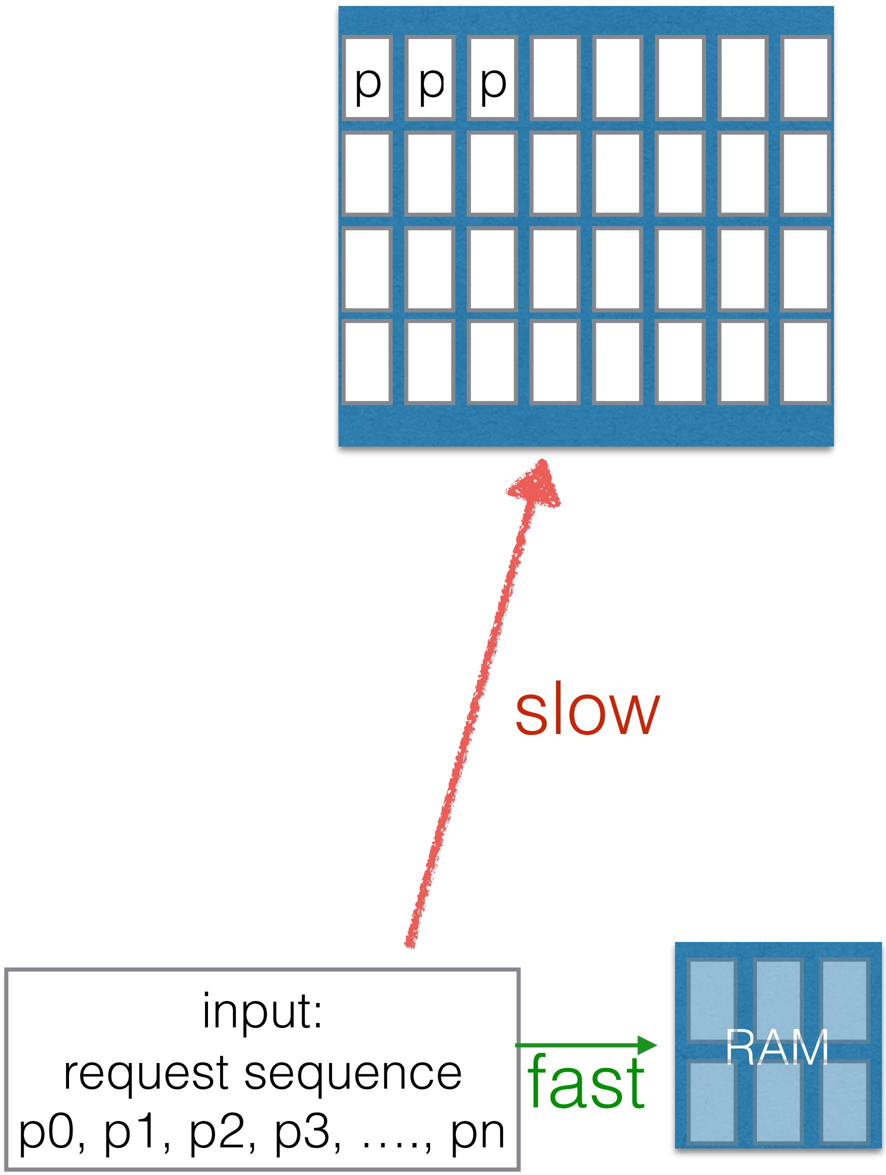


goal

minimize total cost of

bringing pages from disk

Paging problem



input

s = p0, p1, p2, p3, … pn

output

for each pi, which p to evict from cache

goal

minimize # misses

LFD is optimal

- proposed by Belady
- LFD: longest forward distance
- synonyms: OPT, MIN, clairvoyant, Belady’s optimal alg
- proof given in class

LFD proof of optimality

- let ALG be any paging algorithm, s input seq. Will construct a seq of algorithms: ALG = ALG0, ALG1, ALG2, … ALGn = LFD st
- cost(ALG0, s) >= cost(ALG1, s) >= cost(ALG2, s) >= … cost(ALGn) = LFD, n = len(s)
- how? by induction: suppose ALGi is determined
- construct ALG(i+1) as follows: for the first i-1 requests, do the same eviction decisions as ALGi. On request i, ALGi evicts the page currently in its cache that will be request furthest into the future, call this page v. ALG(i) may or may not evict this same page. If it does, let ALG(i+1) = ALG(i) for the future too, so then done.
- Otherwise ALG(i) evicts page u != v. So after processing i, both caches are the same, but ALG(i) has v and ALG(i+1) has u, where u is requested before v.
- On future requests, ALG(i+1) mirrors the same decisions as ALG(i) except if ALG(i) evicts v, then ALG(i+1) evicts u. Since u will be requested first, ALG(i) will incur a miss when u is requested, at which point both may or may not be identical (off by 1). If v is requested in the future, then the costs will be identical.

Belady’s anomaly

- increasing the cache size may result in more misses
- hw: FIFO suffers from this phenomenon
- hw: LRU doesn’t

Competitive analysis

Which replacement policy is better?

$$\textbf{competitive ratio of ALG} = \max_{\text{sequences } s} \frac{\text{misses(ALG, s)}}{\text{misses(OPT, s)}}$$

ALG is said to be **k-competitive** if its competitive ratio is $\leq k$

LRU is k-competitive

for any input sequence s:

misses(LRU, s) $\leq$ k misses(OPT, s)

Proof sketch: divide s into k-phases, where each phase consists of references to exactly k distinct pages
then misses(LRU, s) $\leq$ k
and misses(OPT, s) $\geq$ 1

FIFO is k-competitive

for any input sequence s:

misses(FIFO, s) $\leq$ k misses(OPT, s)

Proof: homework

LFU is not competitive

there exists a sequence of input sequences s1, s2, s3, …

$$\lim_{i \to \infty} \frac{\text{misses(LFU, si)}}{\text{misses(OPT, si)}} = \infty$$

Proof: let si = p0 (x i+1), p1 (x i+1), … p(k-1) (x i+1),
[pk, p(k+1)] (x i)

then misses(LFU, si) = k-1 + 2i

and misses(OPT, si) = k

LIFO is not competitive

there exists a sequence of input sequences s1, s2, s3, …

$$\lim_{i\to\infty} \frac{\text{misses(LIFO, si)}}{\text{misses(OPT, si)}} = \infty$$

Proof: homework

A timeline of paging analysis

- Belady's paper shows that **MIN is optimal**
- **competitive analysis,** Sleator and Tarjan
- 1995: 1st analysis of paging with **locality of reference**, Borodin et al. using access graphs
- 1996: strongly competitiveness, Irani et al.
- 1999: LRU is better than FIFO
- 2000: markov paging, Karlin et al.
- 2007: relative worst order, Boyar et al
- 2009: relative dominance, Dorrigiv et al
- 2010: parameterized analysis, Dorrigiv et al
- 2012: access graph model under worst order analysis, Boyar et al
- 2013: access graph model under relative interval analysis, Boyar et al.
- 2015: competitiveness in terms of locality parameters, Albers et al.

over the years:
- alternative performance measures proposed
- various frameworks for locality of reference proposed

Recap

- LFD, LRU, FIFO, LFU
- provable results:
  - LFD is optimal
  - LRU and FIFO are k-competitive, LFU is not
- practice: LRU-LFU hybrids are preferred because workloads tend to exhibit locality of reference
- practice: clock approximates LRU with low overhead
- practice: best algorithm will depend on the workload

Extra

Locality of reference

An input sequence of references exhibits
locality of reference if
a page that is referenced is likely
to be referenced again in the near future.

Modeling locality of reference

Characteristic vector of an input sequence s

$$C = (c_0, \ldots, c_{p-1})$$

p: number of distinct pages referenced

c_i: number of distance-i requests

distance-i request: a request to a page such that the number of distinct pages requested since the last time this page was requested is i

Ref: Quantifying Competitiveness in Paging with Locality of Reference, Albers, Frascaria, 2015

LRU beats FIFO in the presence of locality of reference

Characteristic vector of an input sequence s

$$C = (c_0, \ldots, c_{p-1})$$

What does the characteristic vector of a sequence with high locality of reference look like?

Ref: Quantifying Competitiveness in Paging with Locality of Reference, Albers, Frascaria, 2015