Cache Algorithms

Online Algorithm

Huaping Wang

Apr.21

Outline

- Introduction of cache algorithms
- A worst-case competitive analysis of FIFO and LRU algorithms
- The randomized marker algorithm
- Competitive analysis for the randomized marker algorithm
- Conclusion

Online algorithm

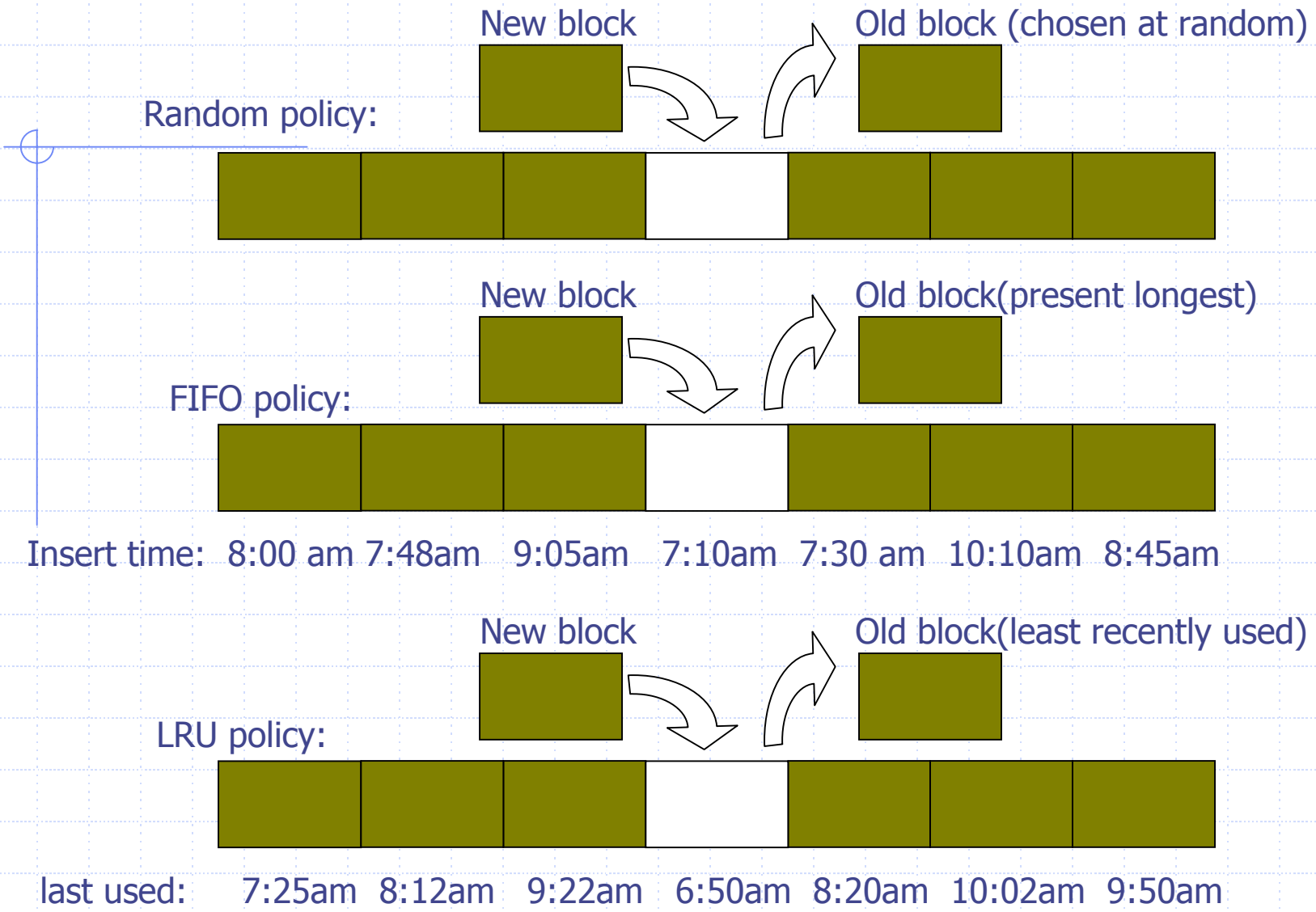
- An ***Online algorithm*** responds to a sequence of ***service requests***,each an associated cost.
- If an algorithm is given the entire sequence of service requests in advance, it is said to be an ***offline*** algorithm.
- We often employ a ***competitive analysis*** to analyze an online algorithm.

Cache memory

- Revisiting information of web pages shows spatial locality and temporal locality.
- Cache memory is used to store copies of web pages to exploit these localities of reference.
- We assume that a web page can be placed in any slot of the cache. This is known as a fully associative cache.

Cache algorithms (Page replacement policies)

- First-in, First-out(FIFO): Evict the page that has been in the cache the longest
- Least recently used (LRU): Evict the page whose last request occurred furthest in the past.
- Random: Choose a page at random to evict from the cache.



The Random,FIFO,and LRU page replacement policies

Complexity of implement(Random)

- It only requires a random or pseudo-random number generator
- Overhead is an O(1) additional amount of work per page replacement
- Makes no attempt to take advantage of any temporal or spatial localities

Complexity of implement(FIFO)

- FIFO strategy just requires a queue Q to store references to the pages in the cache
- Pages are enqueued in Q
- Simply performs a dequeue operation on Q to determine which page to evict.
- This policy requires O(1) additional work per page replacement
- Try to take advantage of temporal locality

Complexity of implement(LRU)

- Implementing the LRU strategy requires the use of a priority queue Q
- When insert a page in Q or update its key, the page is assigned the highest key in Q
- Each page request and page replacement is O(1) if Q is implemented with a sorted sequence based on a linked list
- Because of the constant-time overhead and extra space for the priority Queue Q,make this policy less attractive from a practical point of view

Competitive analysis

- Compare a particular ***online*** algorithm **A** to an optimal ***offline*** algorithm,**OPT**.
- Given a particular sequence P $=(p_1,p_2,\ldots,p_n)$ of service requests,
- Let cost(A,P)denote the cost of A on P
- Let cost(OPT,P) denote the cost of the optimal algorithm on P.

Competitive analysis(cont.)

- The algorithm A is said to be c-**competitive** for P if

  $$cost(A,P) \leq c * cost(OPT,P) + b,$$

  For some constant $b \geq 0$.
- If A is c-competitive for every sequence P, then we simply say that A is c-competitive,and we call c the **competitive ratio** of A.
- If b=0,then we say that the algorithm A has a **strict** competitive ratio of c

A worst-case competitive analysis of FIFO and LRU

- Suppose a cache containing m pages
- Consider the FIFO and LRU methods performing page replacement for a program that has a loop that repeatedly request m+1 pages in a cyclic order
- So for a sequence P =(p_1,p_2,…p_n) of page requests,FIFO and LRU will evict n times

A worst-case competitive analysis of FIFO and LRU(cont.)

- For OPT,is to evict from the cache the page that is requested the furthest into the future
- So the OPT policy will perform a page replacement once every m requests
- Both FIFO and LRU are c-competitive on this sequence P, where

$$c = \frac{n}{n/m} = m$$

The randomized marker algorithm

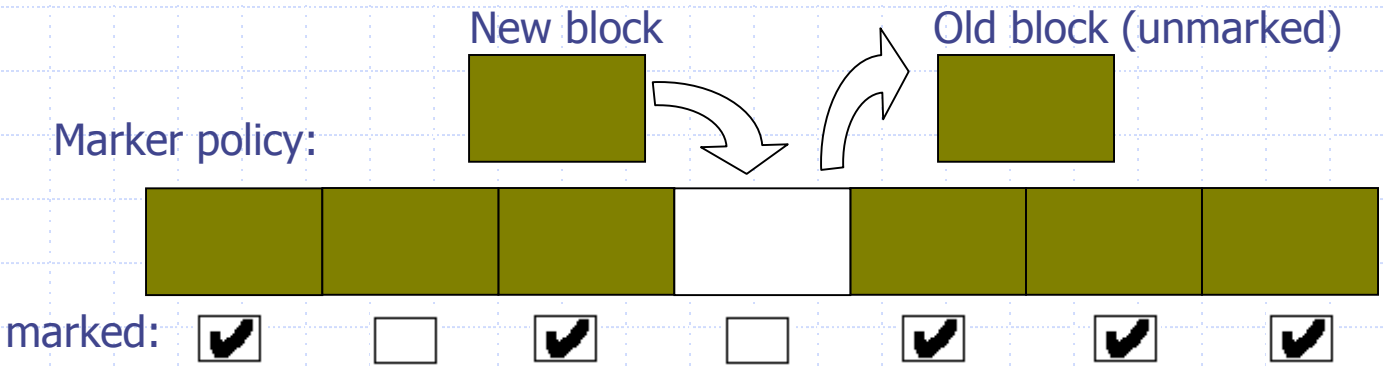
- Emulates the best aspects of the deterministic LRU policy,
- Using randomization to avoid the worst-case situations that are bad for the LRU strategy.

The randomized marker algorithm(cont.)

- Associate,with each page in the cache, a Boolean variable “marked”, which is initially set to “false” for every page in the cache
- If a browser requests a page that is already in the cache, that page’s marked variable is set to “true”.
- Otherwise, if a browser requests a page that is not in the cache, a random page whose marked variable is “false” is evicted and replaced with the new page, whose marked variable is immediately set to “true”

The randomized marker algorithm(cont.)

- If all the pages in the cache have marked variables set to “true”,then all of them are reset to “false”



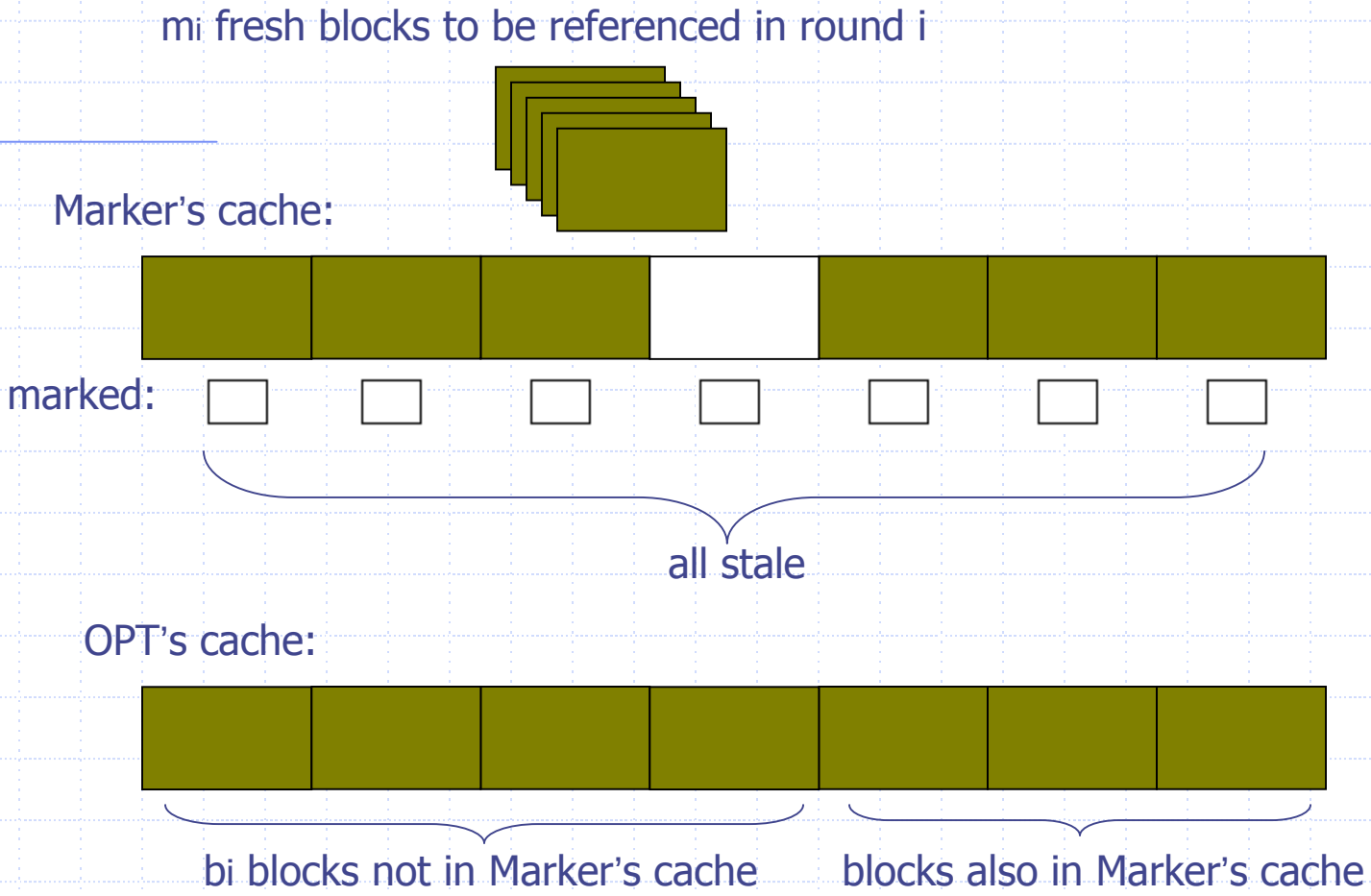
The Marker page replacement policy

Competitive analysis for marker algorithm

- Let P =(p_1,p_2,…,p_n) be a sufficiently long sequence of page requests.
- The Marker policy implicitly partitions the requests in P into rounds
- Each round begins with all the pages in the cache having “false” marked labels
- A round ends when all the pages in the cache have “true”marked labels(with the next request beginning the next round,since the policy then resets each such label to “false”)

Competitive analysis for marker algorithm(cont.)

- Consider the *i* th round in P,and call a page requested in round i ***fresh*** if it is not in the Marker's cache at the beginning of round i
- We refer to a page in the Marker's cache that has a false marked label ***stale***.
- Let m_i denote the number of fresh pages referenced in the *i* th round
- Let b_i denote the number of pages that are in the cache for OPT algorithm at the beginning of round i and are not in the cache for the marker policy at this time



The sate of Marker's cache and OPT's cache at the beginning of round i

Page replacements of OPT

- The algorithm OPT must perform at least

$$\max\{m_i - b_i, b_{i+1}\} \geq \frac{m_i - b_i + b_{i+1}}{2}$$

page replacements in round i

- Summing over all k rounds in P then,

$$L = \sum_{i=1}^{k} \frac{m_i - b_i + b_{i+1}}{2} = (b_{k+1} - b_1)/2 + \frac{1}{2}\sum_{i=1}^{k} m_i$$

Page replacements of marker policy

- The expected number of page replacements performed by the Marker policy is $m_i + n_i$,
- n_i is the expected number of stale pages that are referenced in round i after having been evicted from the cache

Page replacements of marker policy(cont.)

- At the point in round i when a stale page v is referenced, the probability that v is out of the cache is at most f/g
- f is the number of fresh pages referenced before page v
- g is the number of stale pages that have not yet been referenced.

Page replacements of marker policy(cont.)

- The cost to the marker policy will be highest then, if all m_i requests to fresh pages are made before any requests to stale pages
- So the expected number of evicted stale pages referenced in round i can be bounded as follows

$$n_i \le \frac{m_i}{m} + \frac{m_i}{m-1} + \frac{m_i}{m-2} + \ldots + \frac{m_i}{m_i+1} \le m_i \sum_{j=1}^{m} \frac{1}{j}$$

Page replacements of marker policy(cont.)

- Noting that this summation is known as the *m* th harmonic number, which is denoted H_m, we have

$$n_i \le m_i H_m$$

- Thus, the expected number of page replacements performed by the Marker policy is at most

$$U = \sum_{i=1}^{k} (m_i + n_i) = \sum_{i=1}^{k} m_i (H_m + 1) = (H_m + 1) \sum_{i=1}^{k} m_i$$

Competitive analysis for marker algorithm(cont.)

- Therefore, the competitive ratio for the Marker policy is at most

$$\frac{U}{L} = \frac{(H_m + 1)\sum_{i=1}^{k} m_i}{(1/2)\sum_{i=1}^{k} m_i} = 2(H_m + 1)$$

- Using an approximation of H_m, namely that $H_m \le \log m$,the competitive ratio for the Marker policy is at most $2\log m$

Conclusion

- Based on the experimental comparisons, the ordering of policies, from best to worst, is as follows: (1) LRU,(2) FIFO,and(3) Random.
- But LRU still has poor performance in the worst case.
- The competitive analysis shows that the Marker policy is fairly efficient

Thanks