CDA0017: Operating Systems

Donghyun Kang (donghyun@changwon.ac.kr)

NOSLab (https://noslab.github.io)

Changwon National University

과거 기록(History) 사용

- 과거 메모리 접근에 대한 기록 사용

Historical Information	Meaning	Algorithms
recency	The more recently a page has been accessed, the more likely it will be accessed again	LRU
frequency	If a page has been accessed many times, It should not be replcaed as it clearly has some value	LFU

LRU

- Least Recently Used
  - 가장 오랫동안 사용하지 않은 페이지 교체
  - 지역성(Locality) 참고
  - Belady's anomaly 이슈 없음
  - 구현이 어려움
    - 페이지의 접근에 대한 내역을 기록해야함
  - 페이지 접근 빈도는 고려하지 않음
  - 모든 워크로드에 적합하지는 않음

LRU (Cont'd)

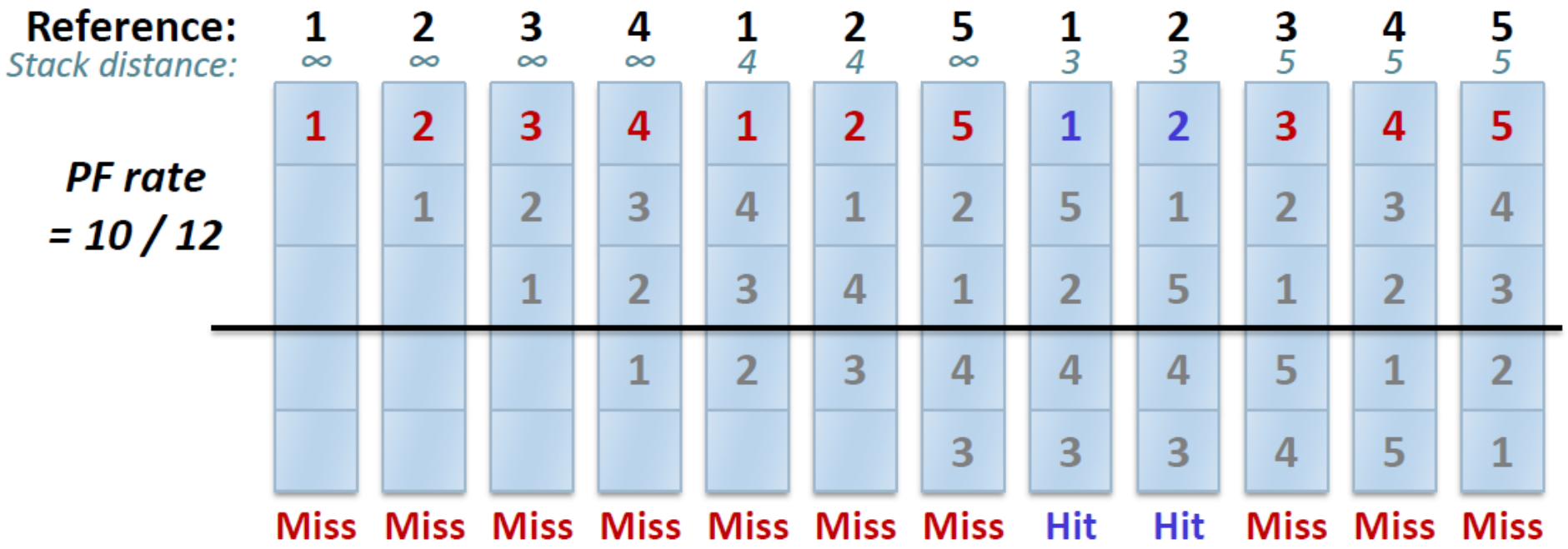
- 예제

Reference Row

0 1 2 0 1 3 0 3 1 2 1

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		1,2,0
1	Hit		2,0,1
3	Miss	2	0,1,3
0	Hit		1,3,0
3	Hit		1,0,3
1	Hit		0,3,1
2	Miss	0	3,1,2
1	Hit		3,2,1

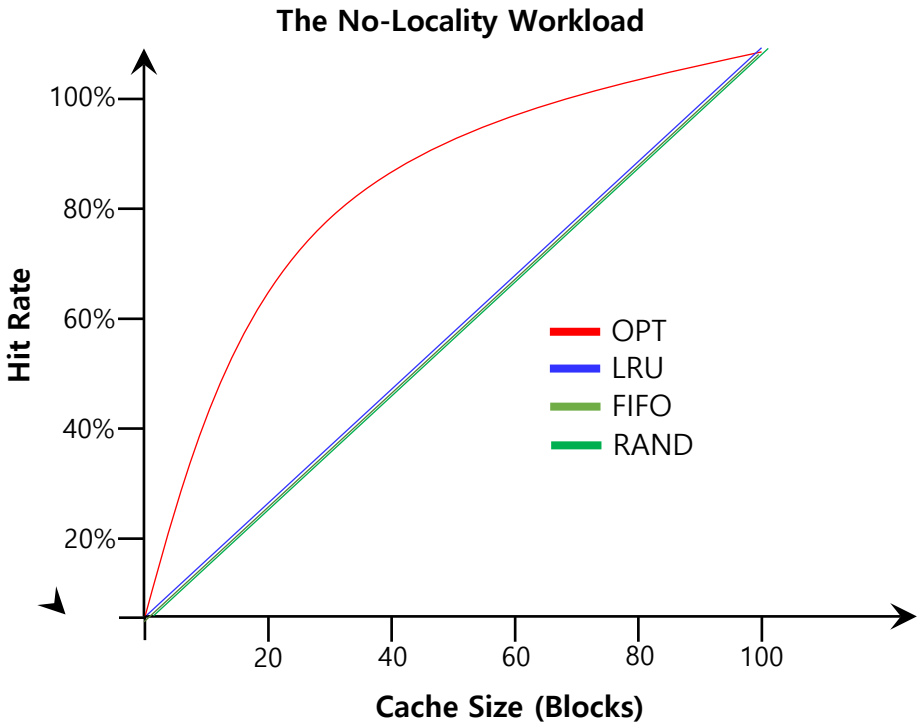
LRU (Cont'd)


Reference:
Stack distance:
1 2 3 4 1 2 5 1 2 3 4 5
∞ ∞ ∞ ∞ 4 4 ∞ 3 3 5 5 5
PF rate
= 10 / 12
Miss Miss Miss Miss Miss Miss Miss Hit Hit Miss Miss Miss

워크로드에 따른 성능 비교

- 무작위로 페이지 접근이 발생함
  - 총 10,000번 페이지 접근 발생
  - 캐시 크기에 따른 성능 비교

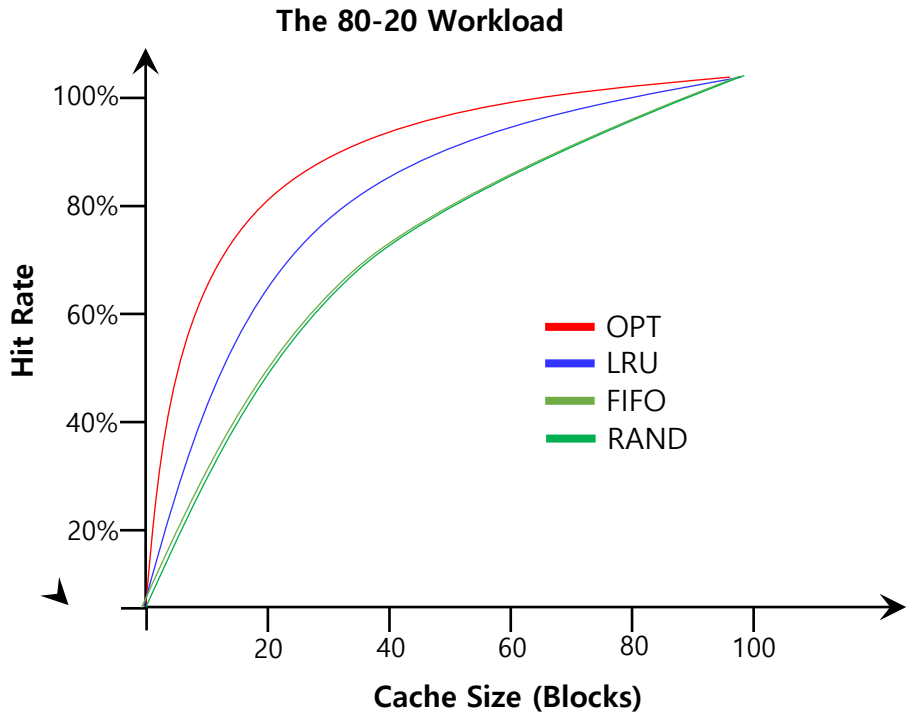
지역성이 없는 경우, 교체 정책의 알고리즘은 Hit Rate에 변화를 주지 않음



“80” 대 “20” 워크로드

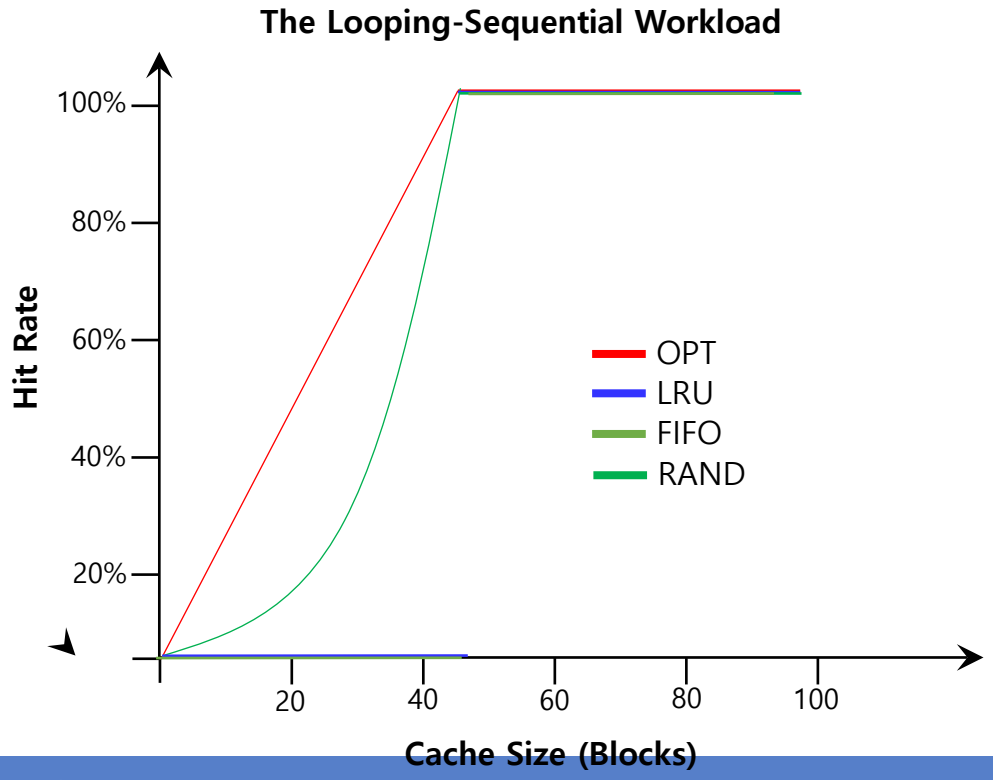
- 인기 있는 페이지 20%: 전체 접근의 80% 접근 발생
- 인기 없는 페이지 80%: 전체 접근의 20% 접근 발생

인기 있는 페이지의 지역성에 따라서 알고리즘의 Hit Rate 차이 발생



순환형 워크로드

- 50 개의 페이지에 대한 순차 접근
  - Starting at 0, then 1, … up to page 49, and then we Loop, repeating those accesses, for total of 10,000 accesses to 50 unique pages.



과거 이력 기반 알고리즘의 구현

- 과거 이력 기반 알고리즘의 성능은 모든 메모리 접근에 대한 이력을 가지고 있어야 함
  - FIFO 방식에 비해 구현이 복잡함
  - 하드웨어의 도움 필요

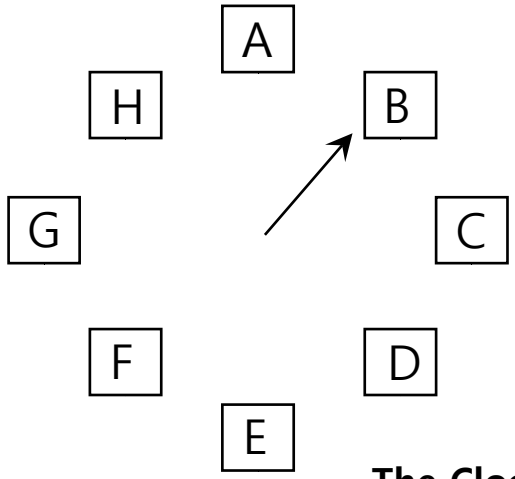
LRU 정책 근사하기

- 하드웨어의 도움으로 use bit을 설정함
  - 페이지가 접근할 때 하드웨어가 1로 설정
  - 하드웨어가 아닌 운영체제에 의해 0으로 설정

- Clock 알고리즘
  - 순환 리스트 형태로 페이지 관리
  - 시계 바늘(Clock hand)을 기준으로 추출(evict) 페이지 선정

Clock Algorithm

- Use bit이 0인 페이지를 찾음
- Use bit이 1인 경우, 0으로 변경하고 바늘(hand)을 이동 시킴

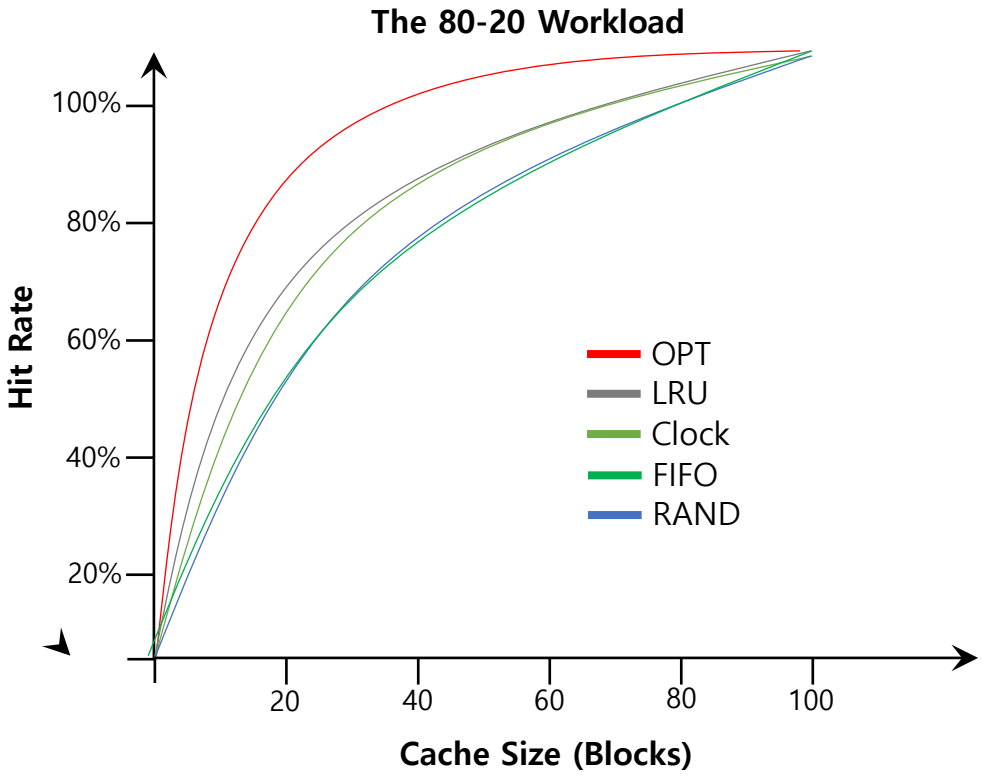


Use bit	Meaning
0	Evict the page
1	Clear **Use bit** and advance hand

The Clock page replacement algorithm

Clock 알고리즘 실험

- Clock algorithm doesn't do as well as perfect LRU, it does better then approach that don't consider history at all.

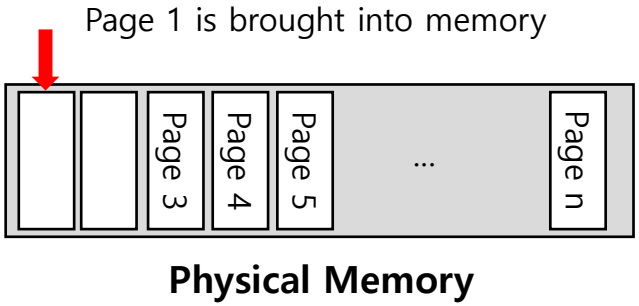


갱신된 페이지(Dirty Page) 고려

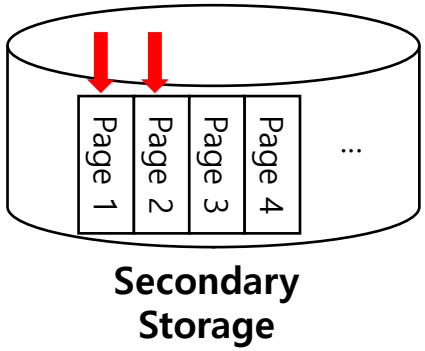
- The hardware include a modified bit (a.k.a dirty bit)
  - Page has been **modified** and is thus **dirty**, it must be written back to disk to evict it.
  - Page has not been modified, the eviction is free.

Prefetching

- The OS guess that a page is about to be used, and thus bring it in ahead of time.

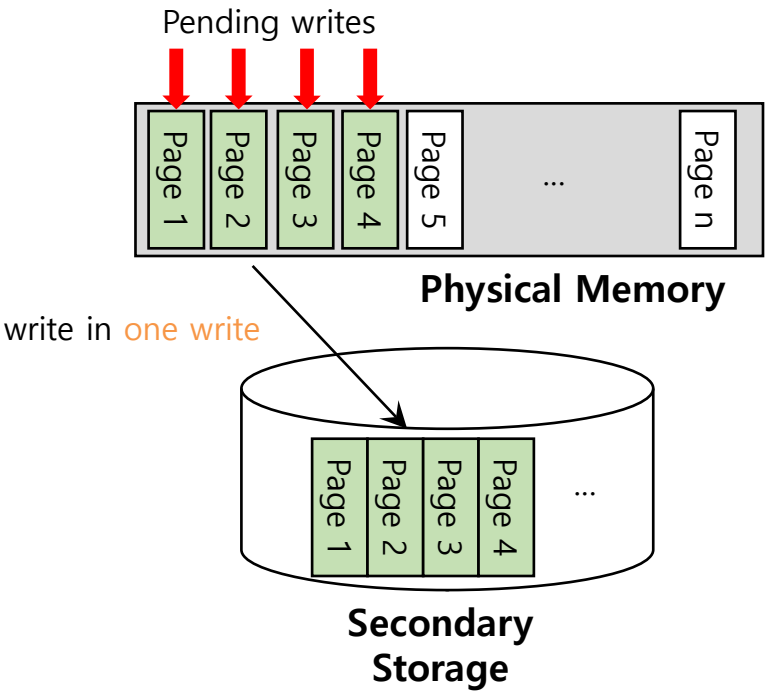


Page 2 likely soon be accessed and
thus should be brought into memory too



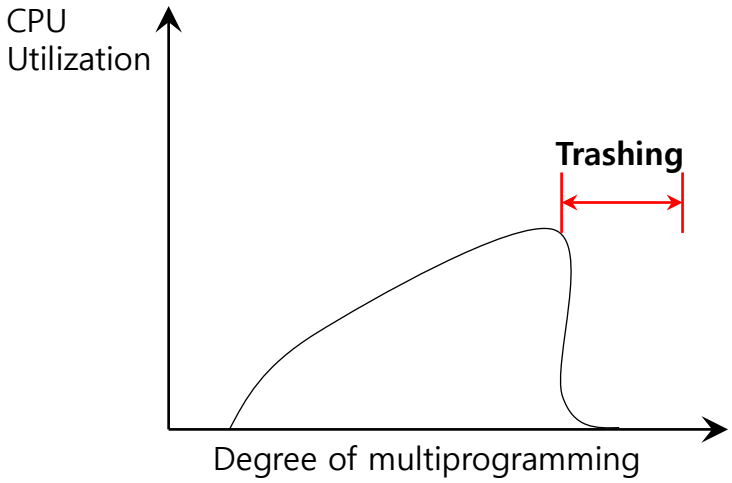
Clustering, Grouping

- Collect a number of pending writes together in memory and write them to disk in one write.
  - Perform a **single large write** more efficiently than **many small ones**.



Thrashing

- Memory is oversubscribed and the memory demands of the set of running processes exceeds the available physical memory.
  - Decide not to run a subset of processes.
  - Reduced set of processes working sets fit in memory.



Q&A