LOBI: A swap scheme for flash memory storages

명지대학교 류연승

2008. 4. 25

NVRAMOS '08

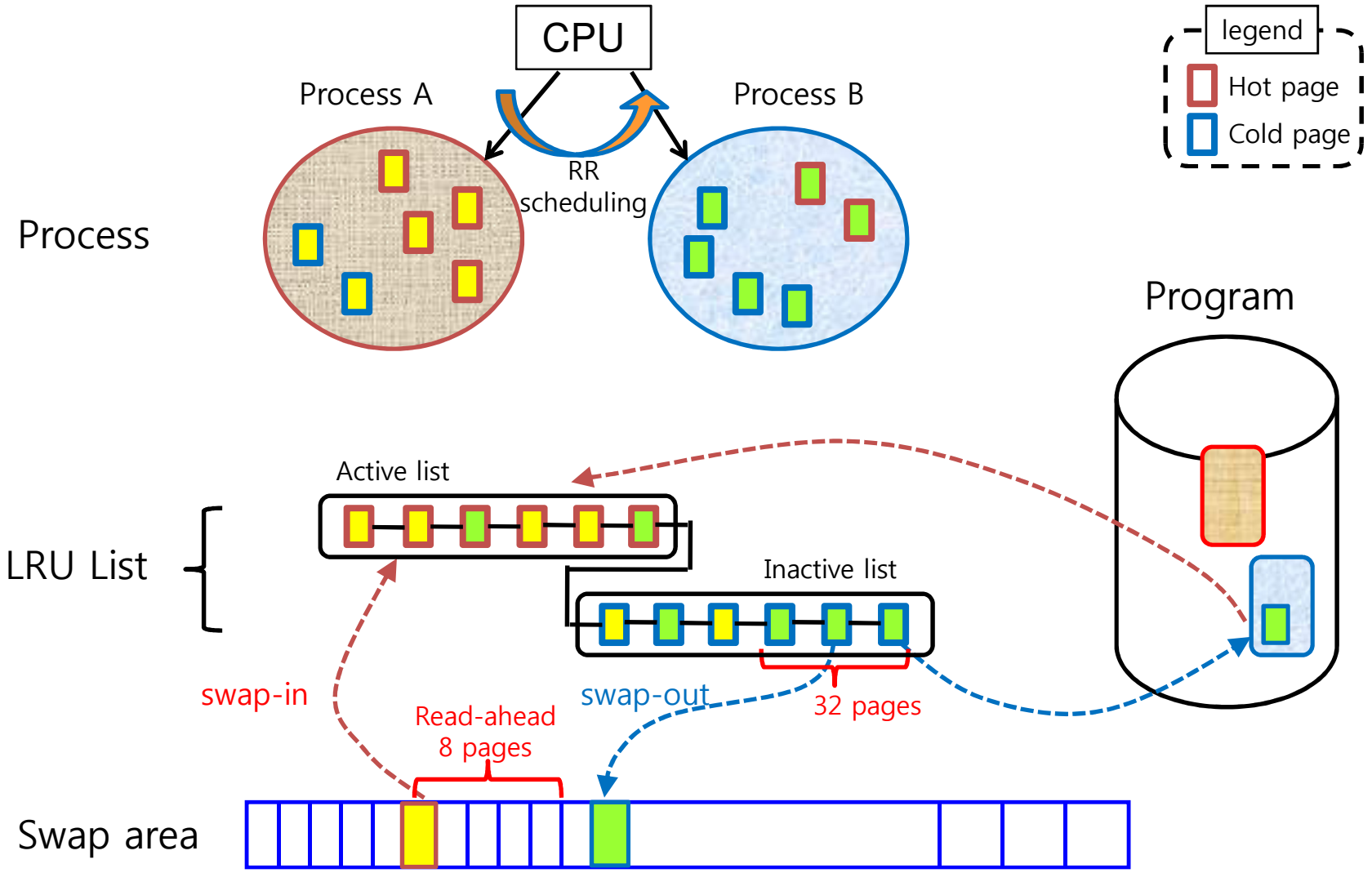
목차

연구 목적

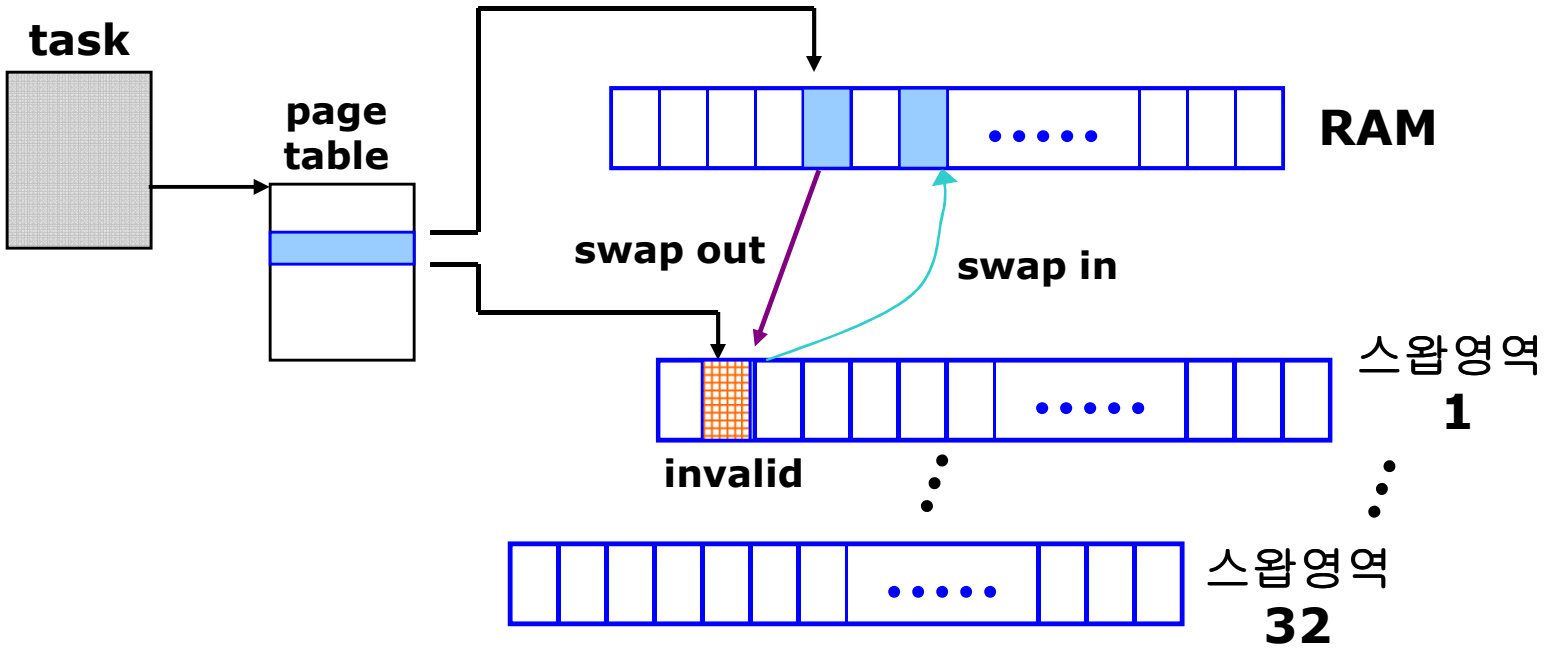
- 기존 리눅스 스왑 시스템의 문제
  - ✓ 하드디스크 특성을 고려하여 seek-time을 최소화
  - ✓ 플래시 메모리의 특성을 고려하지 않음
    - ✓ Seek-time이 없다
    - ✓ Erase before update
    - ✓ Erase 횟수의 제한 (100,000 ~ 1,000,000번)
    - ✓ 느린 Erase 연산 시간 (1~2 ms)

- 플래시 메모리의 특성을 고려하는 스왑 시스템 연구
  - ✓ 가비지 수집 비용 최소화
  - ✓ 마모도 평준화

Overall architecture



Overall architecture



주요 특징

- 스왑 영역은 파일 또는 파티션 형태
- 스왑 영역은 고정 크기의 스왑 슬롯 **(4KB)**으로 구성
- 다중 스왑 영역 **(32**개 까지**)**

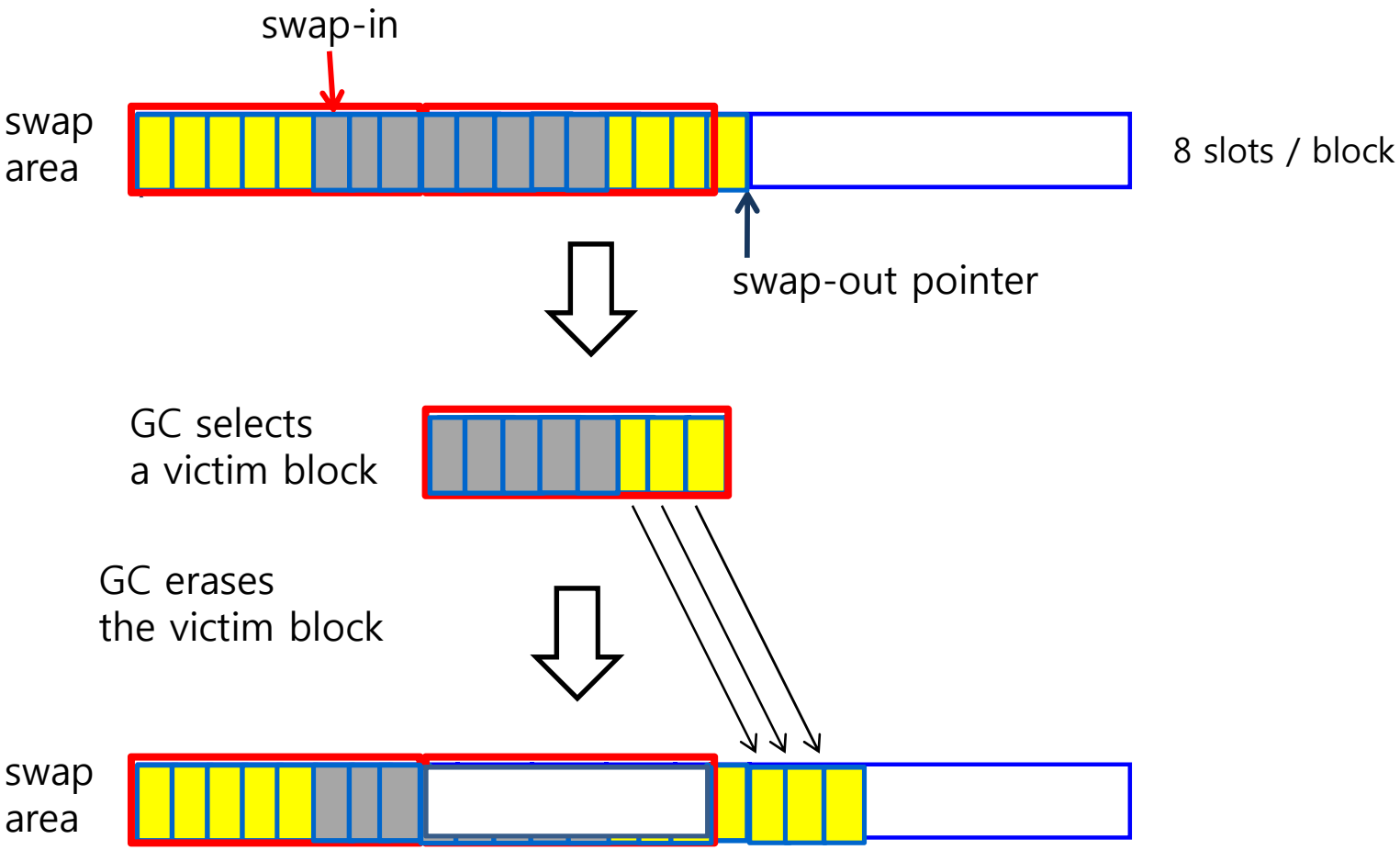
Swap slot allocation/release

- Swap slot allocation
  - kernel stores swap-out pages in contiguous slots to minimize disk seek time when accessing the swap area.
  - It always starts from the last allocated page slot unless one of these conditions occurs:
    - The end of the swap area is reached.
    - SWAPFILE_CLUSTER (usually 256) free page slots were allocated after the last restart from the beginning of the swap area.

- Swap slot release
  - Swap-in 요청에 대하여 최대 8개의 연속된 slot의 페이지를 read-ahead

Garbage collection

- Swap-in에 의해 생겨나는 invalid 슬롯을 free 슬롯으로 바꿔야 함



Cost of garbage collection

- GC cost
  - ✓ Victim 블럭 내의 valid slot들을 다른 블록으로 이동
    - – Valid slot을 read, write
  - ✓ Victim 블록을 erase

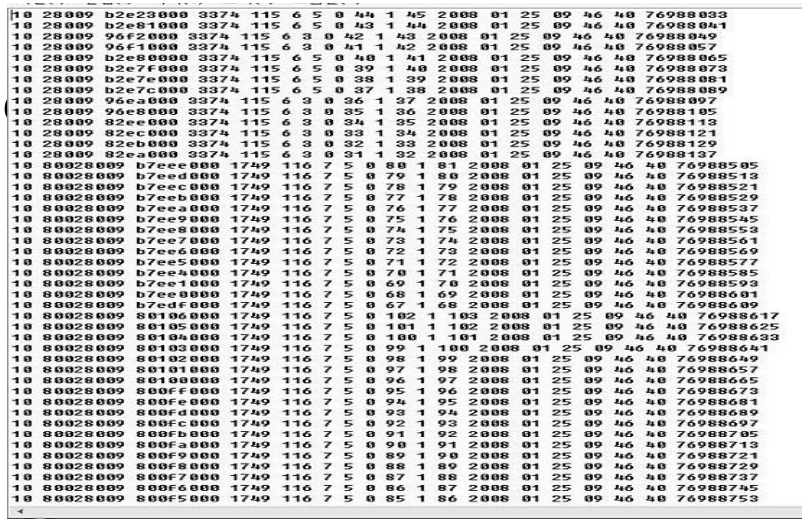
- GC cost를 줄이려면
  - ✓ Victim 블록이 invalid slot을 많이 가져야 함
    - – 데이터 이동 최소화
    - – Free space 확보를 최대화 -> GC 수행 횟수 줄임 -> erase 횟수 줄임
  - ✓ 이를 위해서, 플래시 파일 시스템에서는 hot 데이터를 같은 블록에 저장
    - – 블록 내의 데이터들이 비슷한 시기에 invalid됨

Questions

- 스왑 시스템에서 hot 데이터를 찾아내서 한 블록에 저장할 수 있는가?
- 즉, 페이지가 스왑 아웃될 때 스왑 인되는 시간을 미리 알아내서 분류 저장할 수 있는가?
- 즉, 비슷한 시기에 스왑 인되는 페이지끼리 한 블록에 저장할 수 있는가?

스왑 트레이스 분석

- 리눅스 커널 코드 수정
  - ✓ swap_writepage(), swap_readpage(), blk_add_trace(), etc
- blktrace 프로그램 코드 수정
- 트레이스 데이터
  - ✓ swap in/out flag
  - ✓ swap entry : swap area number, swap slot number
  - ✓ page descriptor의 logical address
  - ✓ page의 owner process's pid
  - ✓ page의 owner process's priority
  - ✓ page의 owner process's time_sli
  - ✓ page type (heap, stack, shared,
  - ✓ shared process counter



스왑 트레이스 분석

- 트레이스 수집을 위한 실험 환경
  - ✓ Desktop PC
    - RAM : 256MB
    - Swap partition : 512MB (= 128 K slots)
    - 커널 컴파일, 프로그램 download 및 컴파일 설치, web surfing, 동영상 play

스왑 트레이스 분석

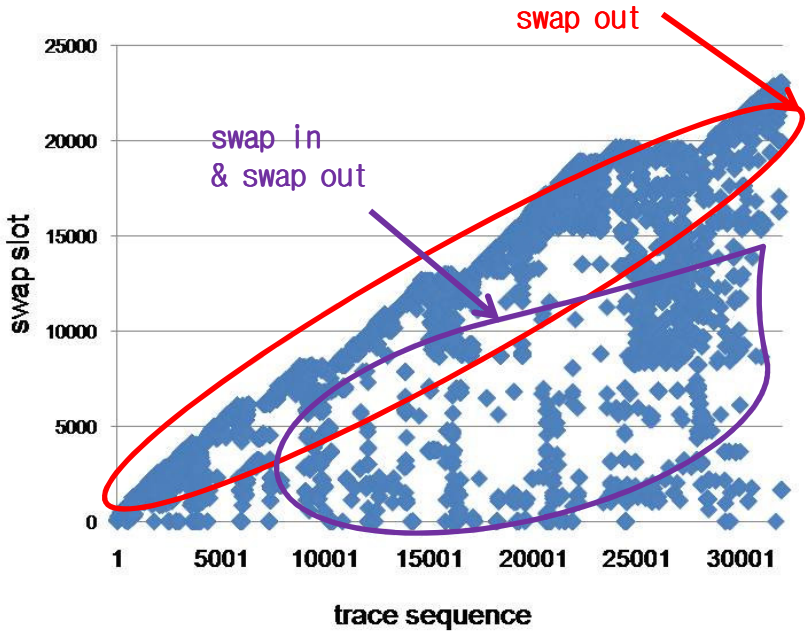
- 트레이스 개요
  - ✓ 스왑 개수

	count	%
swap-out	63,505	62.7
swap-in	37,916	37.3
total	101,421	100.0

  - ✓ 프로세스 개수

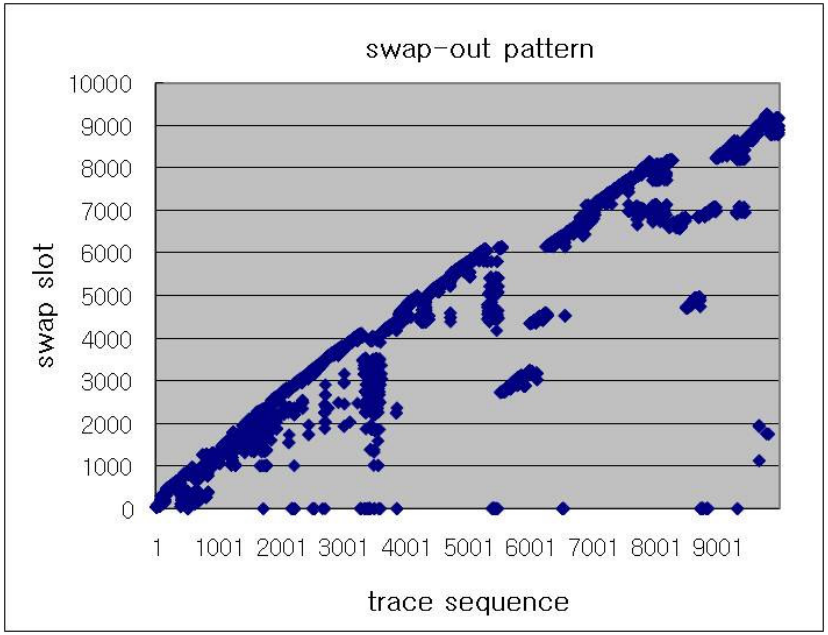
	count
Process	127

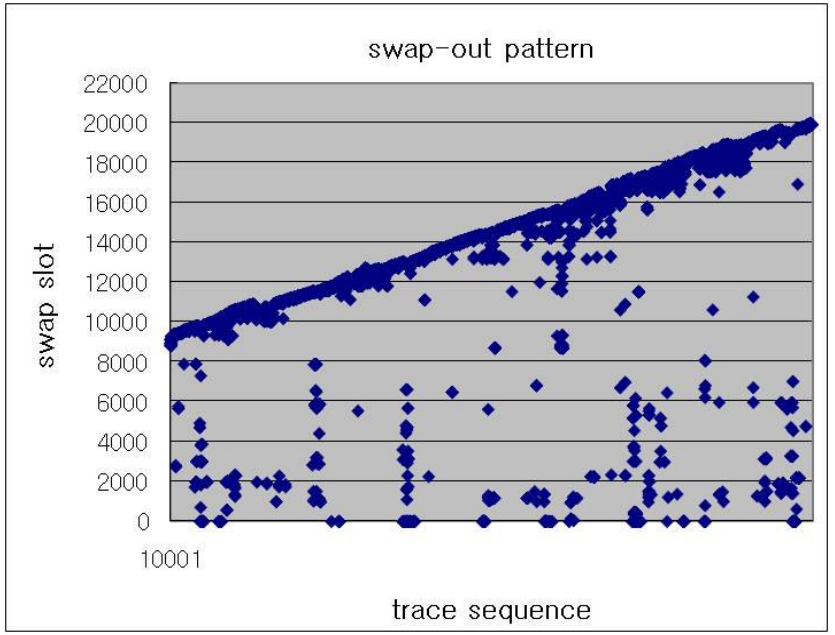
- 스왑이 시작되면 스왑 슬롯을 순차적으로 할당
- page fault에 의해 스왑 인이 발생
- 점차 스왑 영역의 전반에 걸쳐 스왑 아웃과 스왑 인이 발생



스왑 트레이스 분석: slot usage

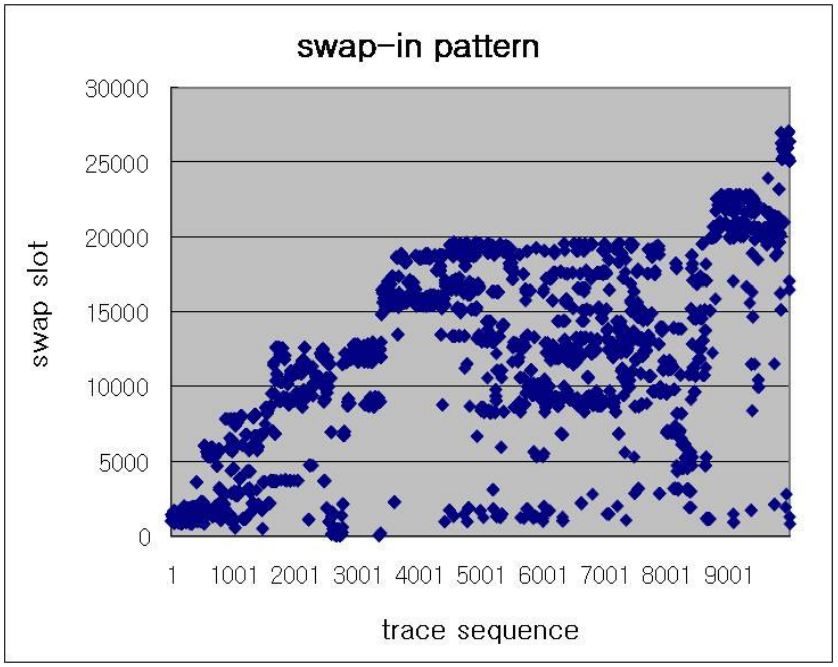
- 스왑 아웃
  - ✓ 순차적으로 슬롯을 할당하여 사용하다가, 스왑인되어 비는 슬롯들도 사용하게 된다.

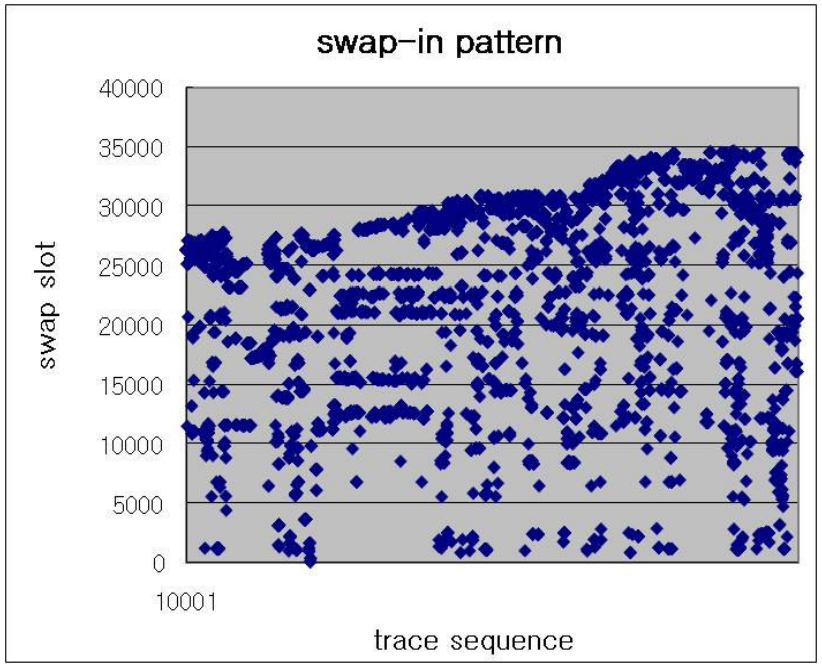




스왑 트레이스 분석: slot usage

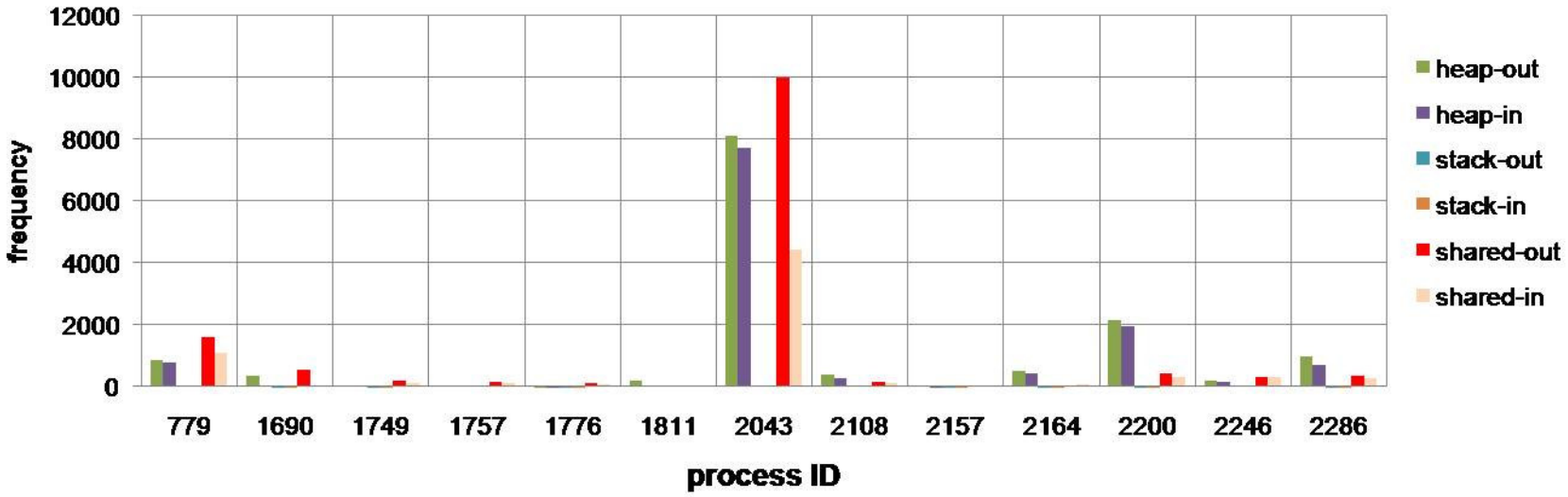
- 스왑 인
  - ✓ 스왑 영역의 전반에 발생하고 있다.





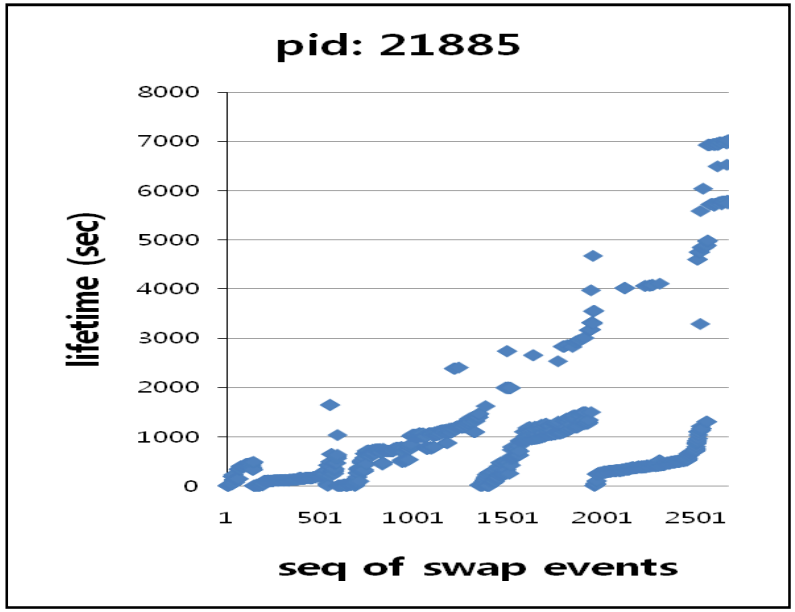
스왑 트레이스 분석 : 프로세스

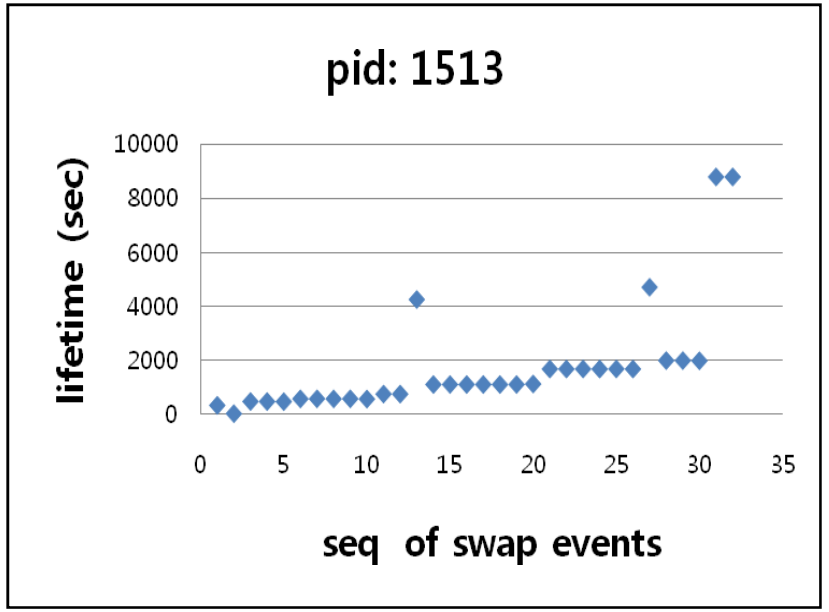
- 프로세스 분석
  - ✓ 특정 소수의 프로세스들이 많이 스왑된다.



스왑 트레이스 분석 : lifetime

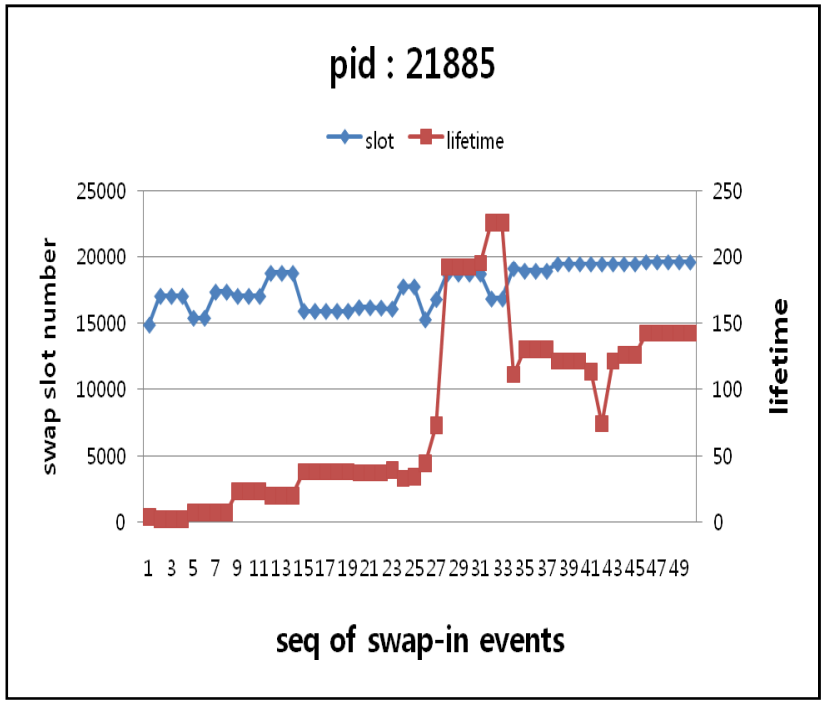
- 한 프로세스에서 수행된 스왑 페이지의 lifetime 분석
  - ✓ 페이지들의 lifetime은 다양하다.

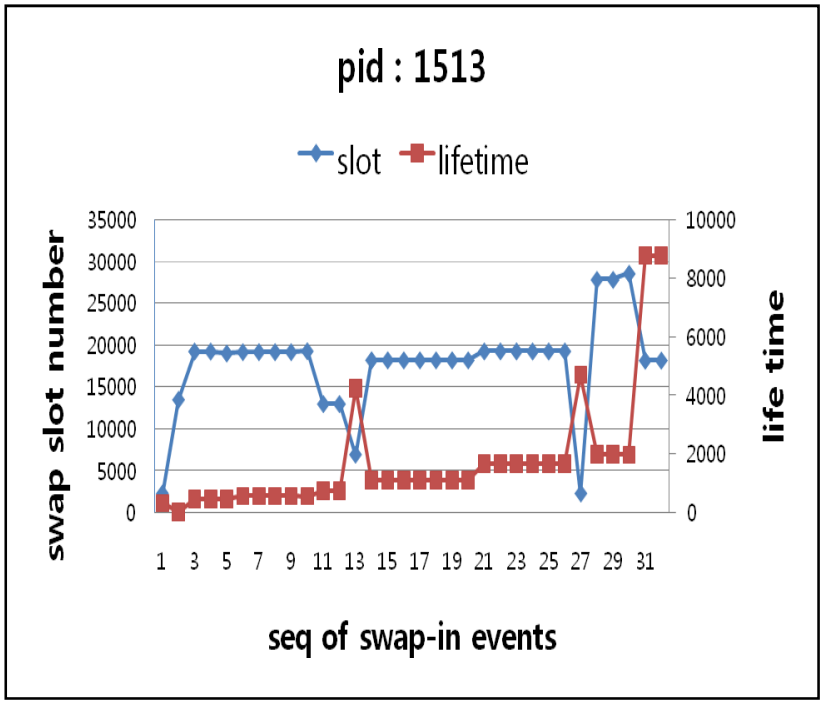




스왑 트레이스 분석 : lifetime

- 스왑 슬롯과 lifetime간의 관계 분석
  - ✓ 연속된 슬롯에 저장되어 있는 페이지들이 동시에 스왑 인 되고 있으며 이들의 lifetime이 비슷함을 알 수 있다





스왑 시스템 설계

- 고려사항

스왑 워크로드 특성

- 동일한 프로세스의 동일 유형 페이지들이 비슷한 시기에 그룹으로 스왑 아웃됨
- 프로세스별로 스왑 횟수의 차이가 크다

- 프로세스별로 따로 저장
- 스왑 아웃 페이지들을 순차적으로 저장

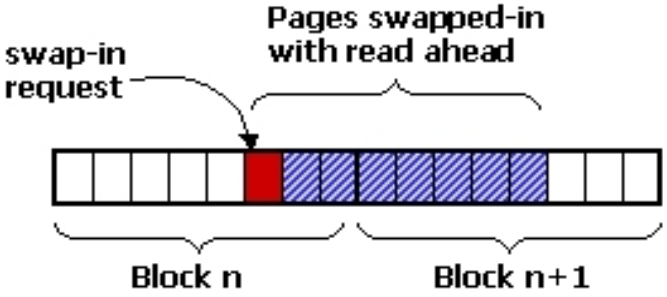
- 비슷한 시기에 스왑 인되어 유사한 lifetime을 가짐

- 여러 개 페이지를 선반입

LOBI (Log-structured swap-Out, Block-aligned swap-In)

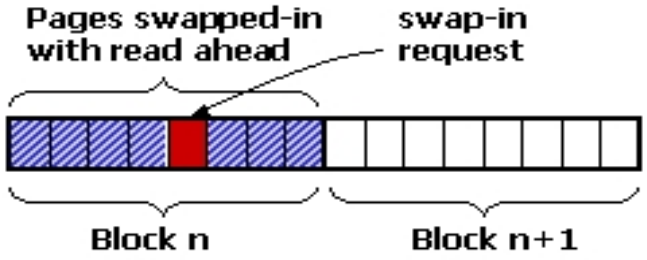
- 로그 구조(log-structured) 슬롯 할당
  - ✓ 스왑 아웃되는 페이지들을 로그 방식으로 저장

- 플래시 삭제 블록을 고려한 선반입 스왑-인
  - ✓ 스왑-인 요청에 대해 해당 페이지가 속한 삭제 블록 내의 페이지들을 선반입



< 리눅스의 선반입 (최대 8페이지) >

두 개 블록에 걸쳐 선반입을 수행

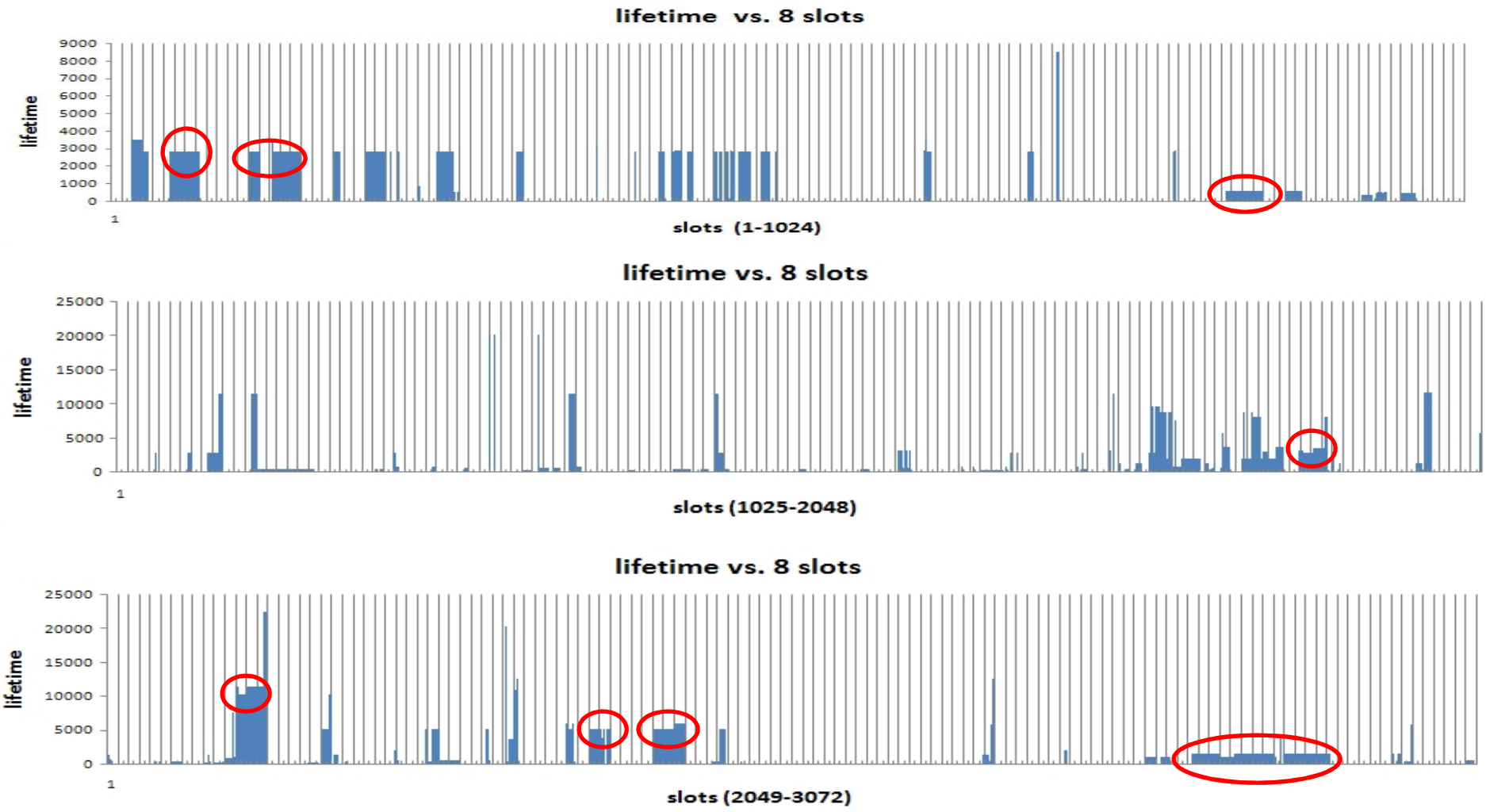


< 삭제 블록 기반 선반입 >

한 블록 내에서만 선반입을 수행

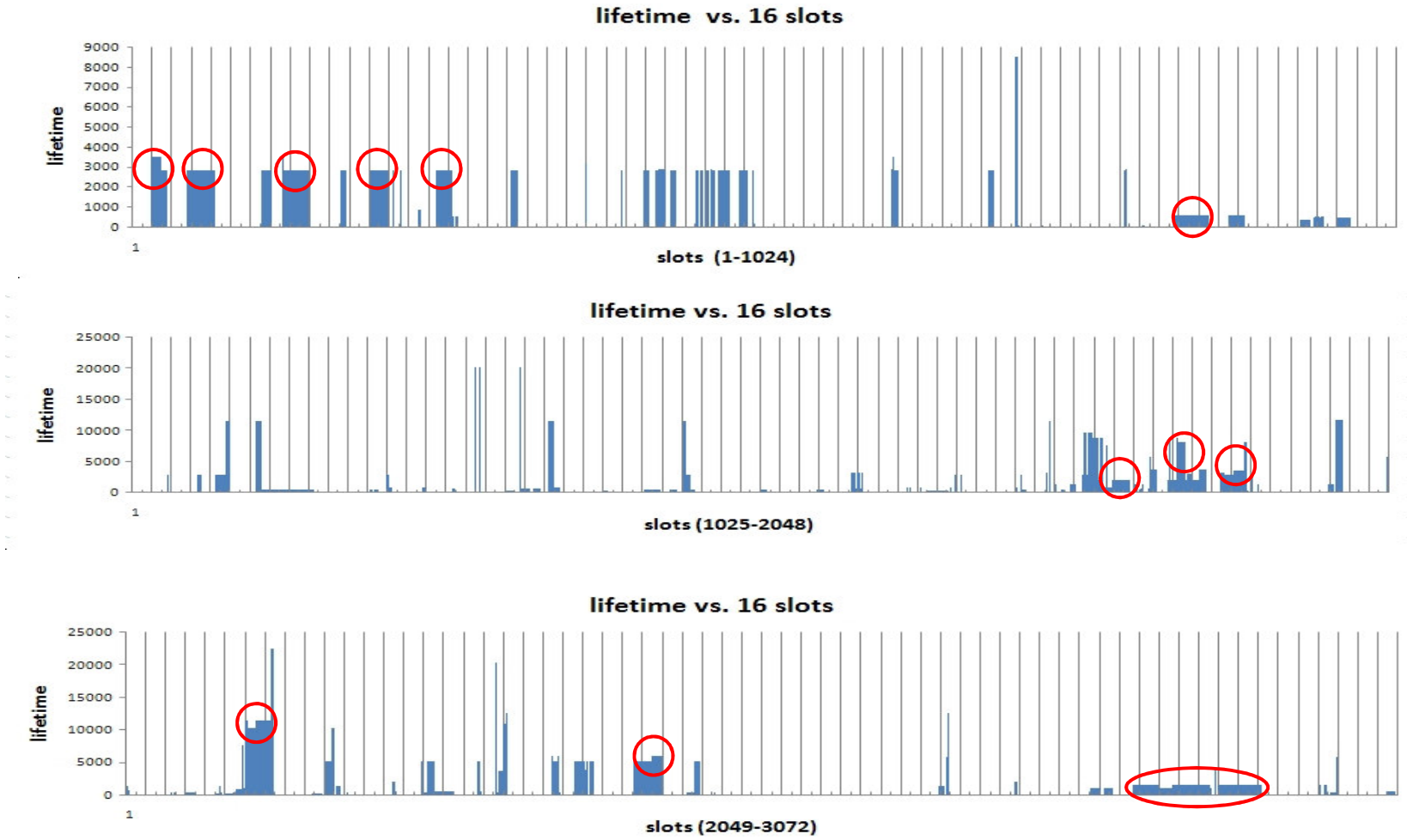
스왑 시스템 설계

- 스왑 아웃되는 페이지를 로그 구조 방식으로 저장했을 때 8개 slot (1/4 block)씩 lifetime을 분석



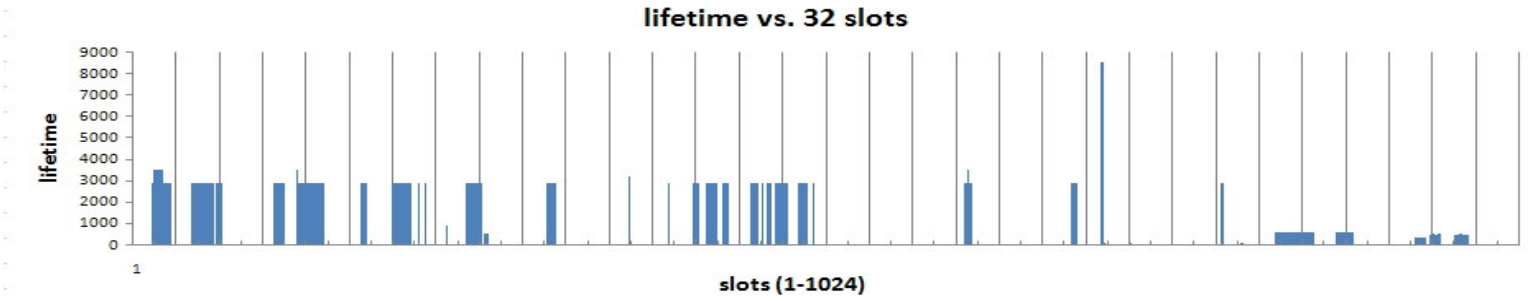
스왑 시스템 설계

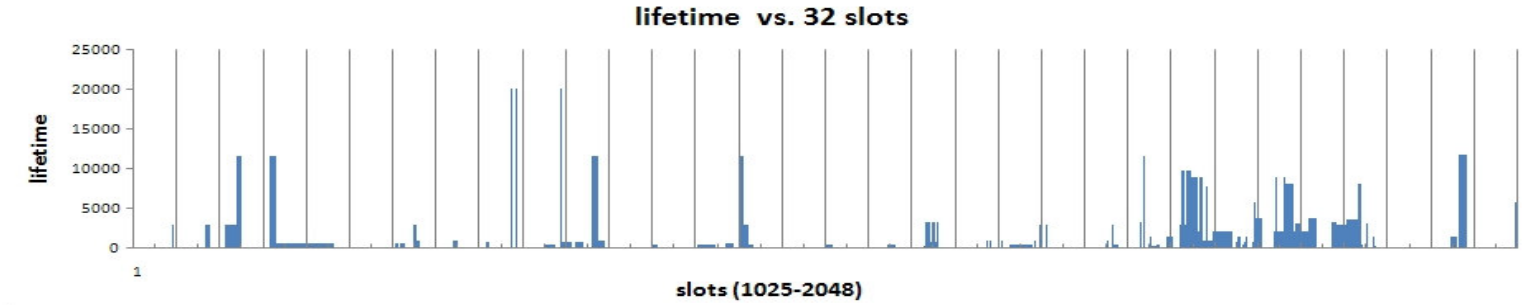
- 스왑 아웃되는 페이지를 로그 구조 방식으로 저장했을 때 16개 slot (1/2 block)씩 lifetime을 분석

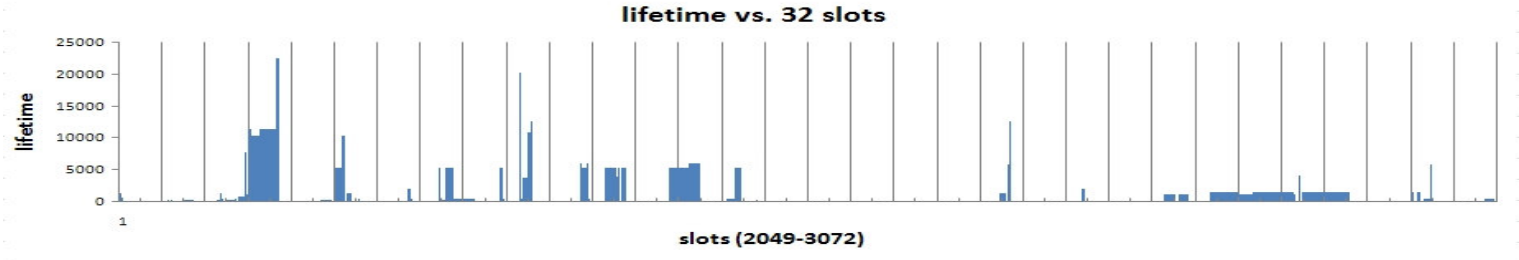


스왑 시스템 설계

- 스왑 아웃되는 페이지를 로그 구조 방식으로 저장했을 때 32개 slot (1 block)씩 lifetime을 분석







성능평가

- Trace-driven simulation

- Linux swap scheme
- LOBI
  - ✓ Block-aligned 8 pages
  - ✓ Block-aligned 16 pages
  - ✓ Block-aligned 32 pages

- Garbage collection algorithm
  - ✓ Greedy (GR)
  - ✓ Cost-Benefit (CB)
  - ✓ Cost-Age-Time with Age (CATA) : victim blocks 1~4
  - ✓ Cost Benefit with Age (CBA) : victim blocks 1~4

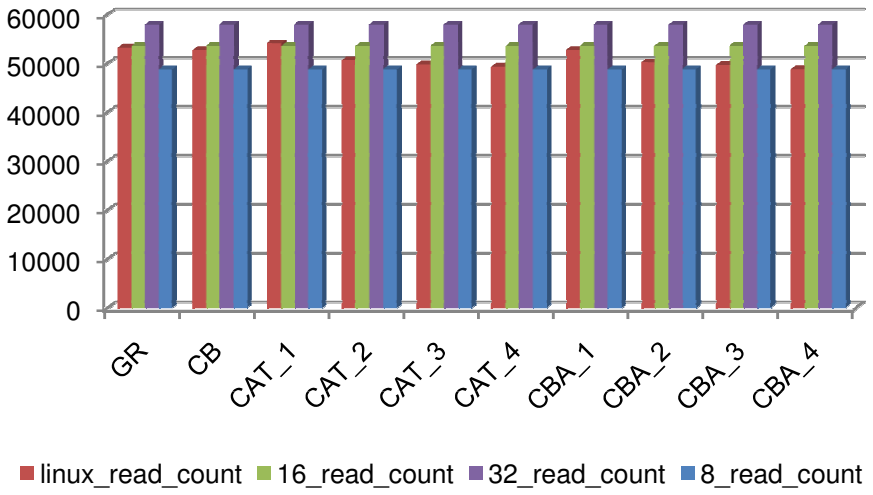
성능평가

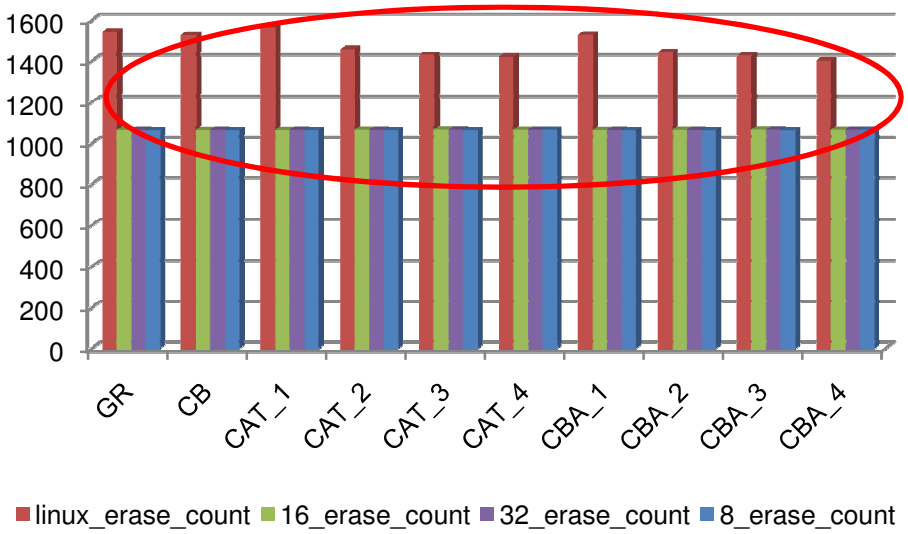
- Swap area size
  - ✓ 1024 blocks (128MB) : 2 rotations with traces
  - ✓ 1536 blocks
  - ✓ 1792 blocks
  - ✓ 2048 blocks (256MB) : 1 rotation

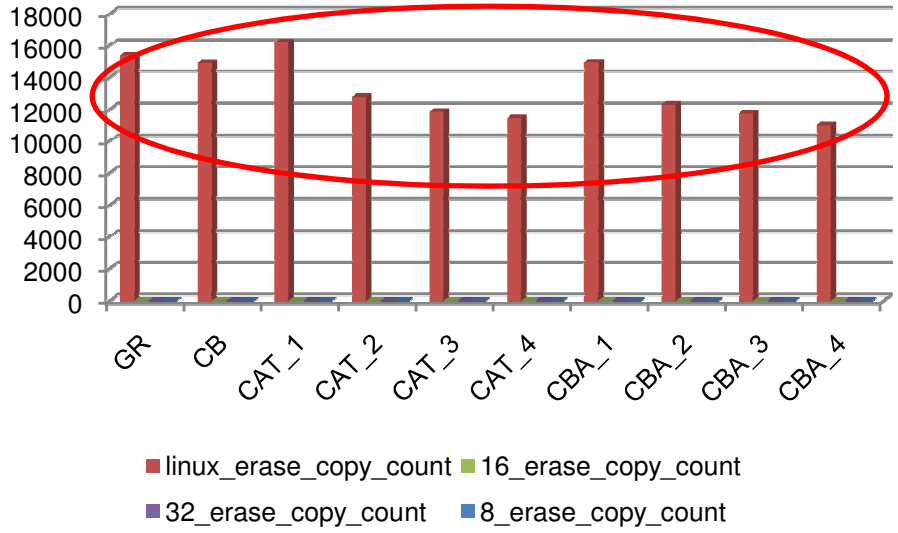
- Performance metric
  - ✓ read count
  - ✓ erase count
  - ✓ erase copy count
  - ✓ wear-leveling degree
  - ✓ garbage collection cost

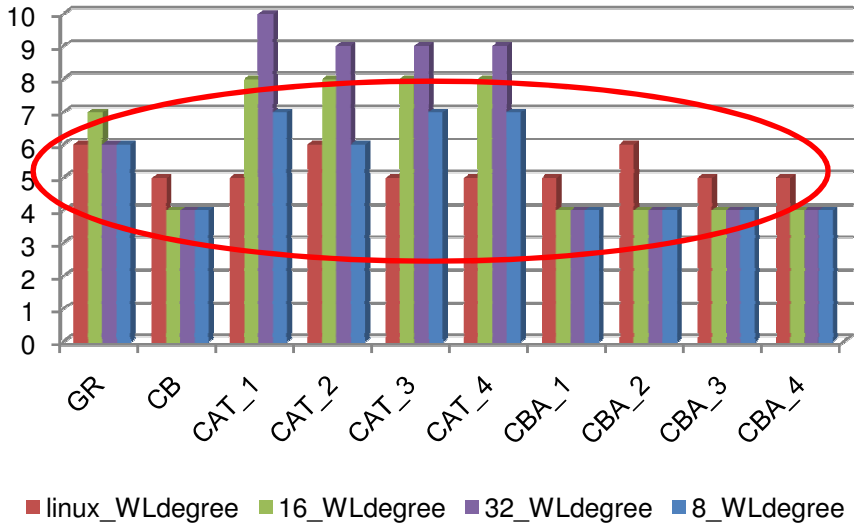
성능평가 결과

- 1024 blocks



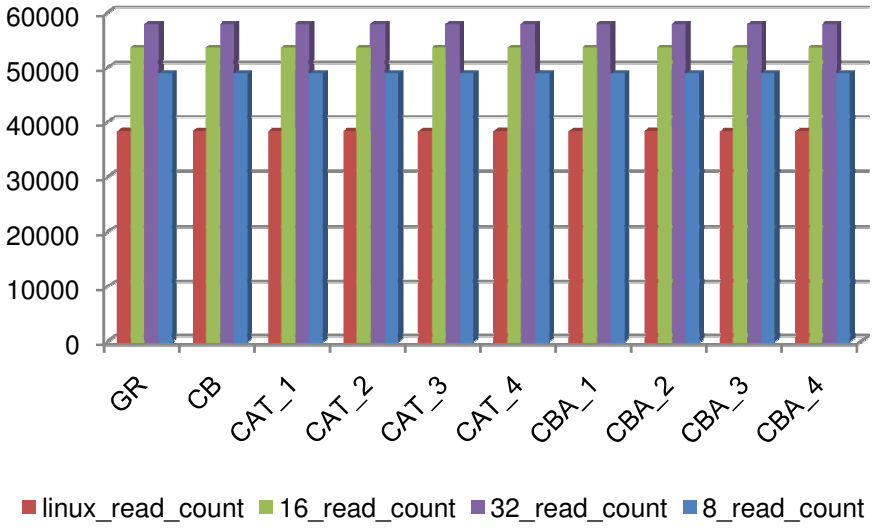


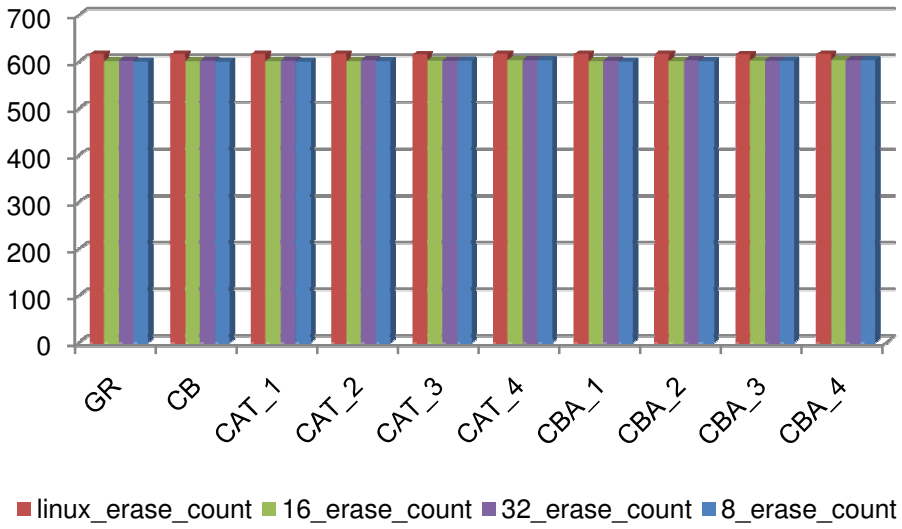


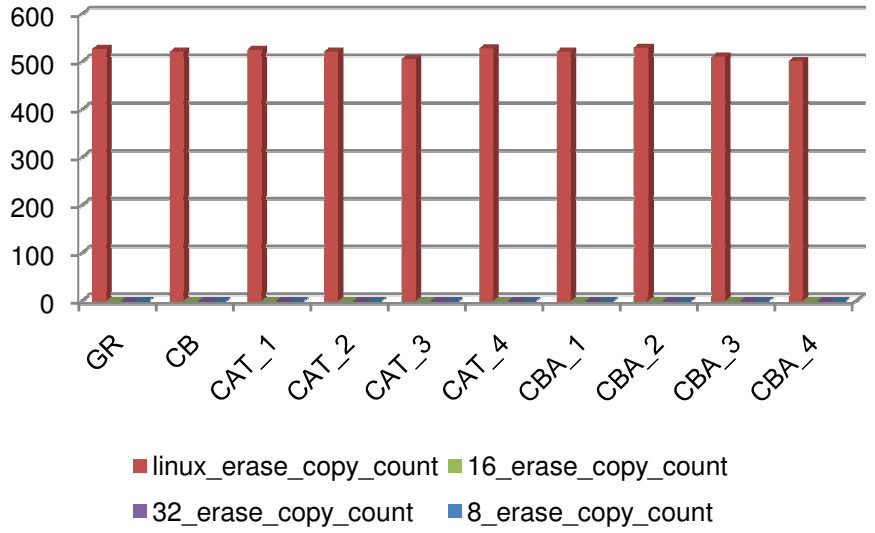


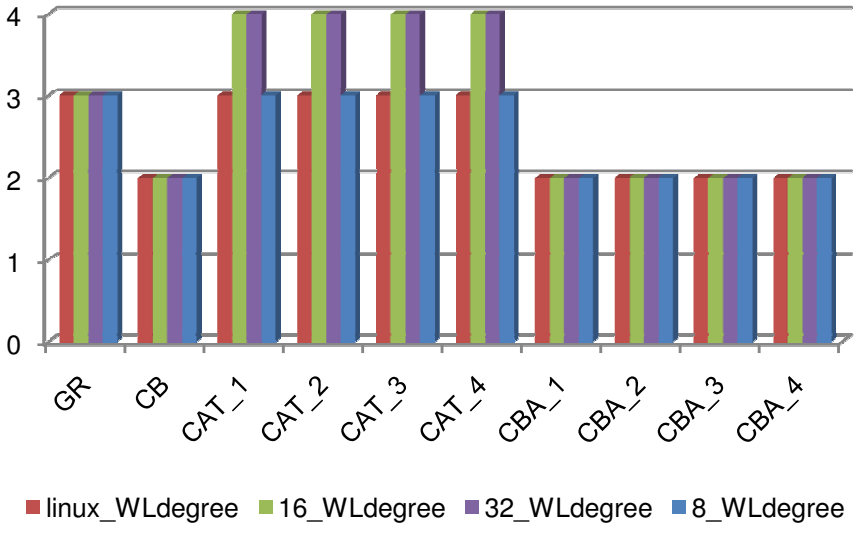
성능평가 결과

- 1536 blocks



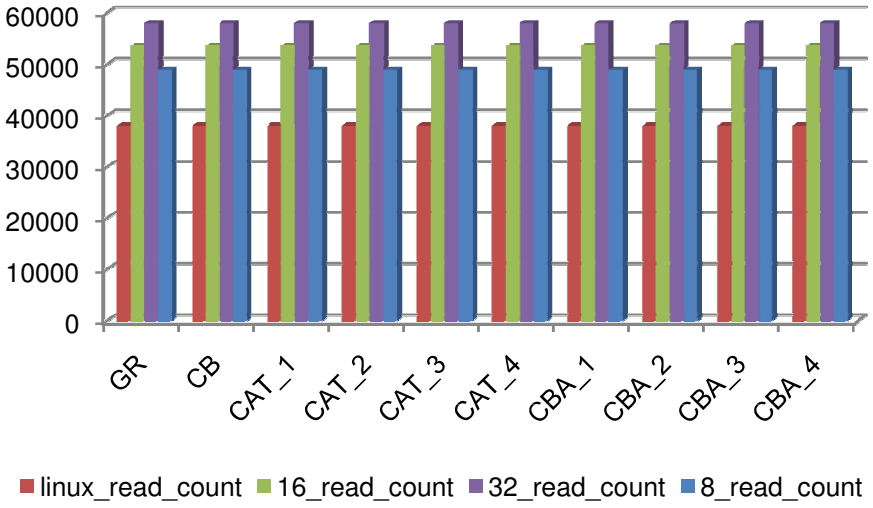


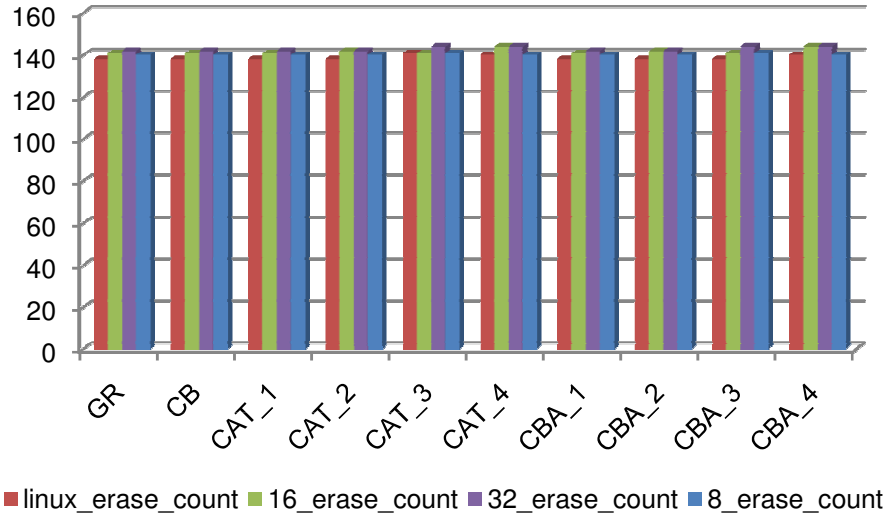


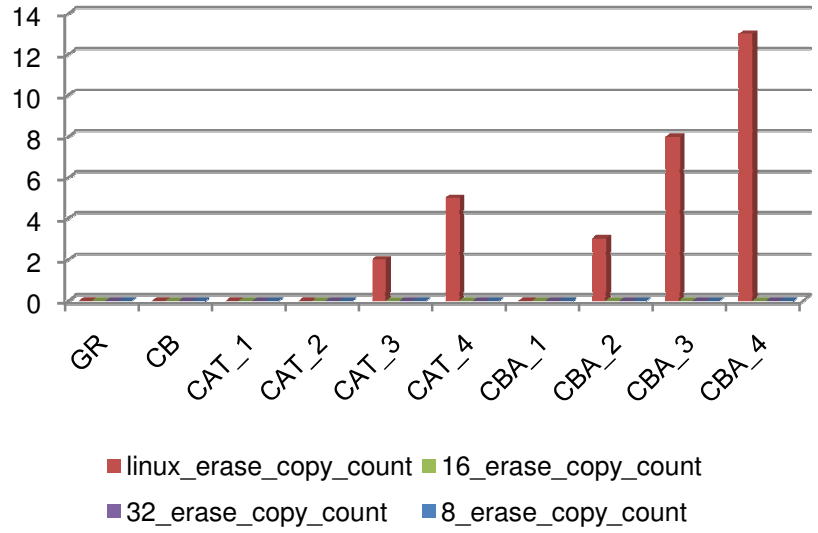


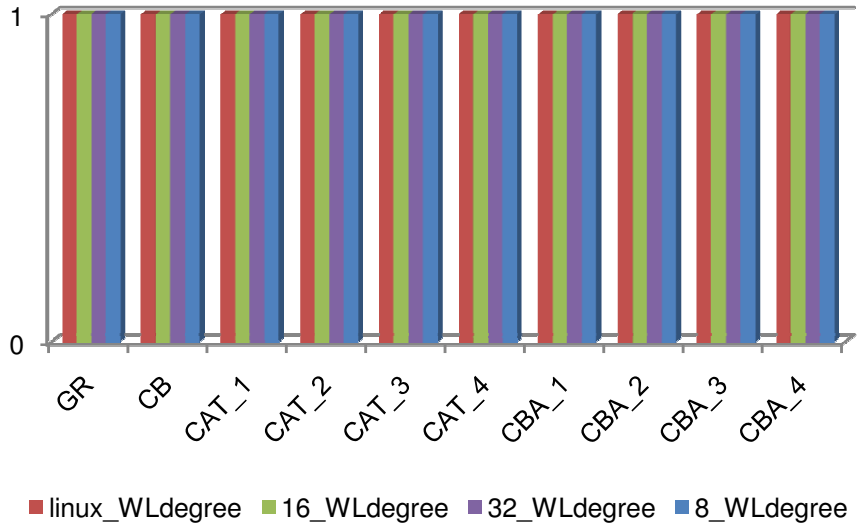
성능평가 결과

- 2048 blocks





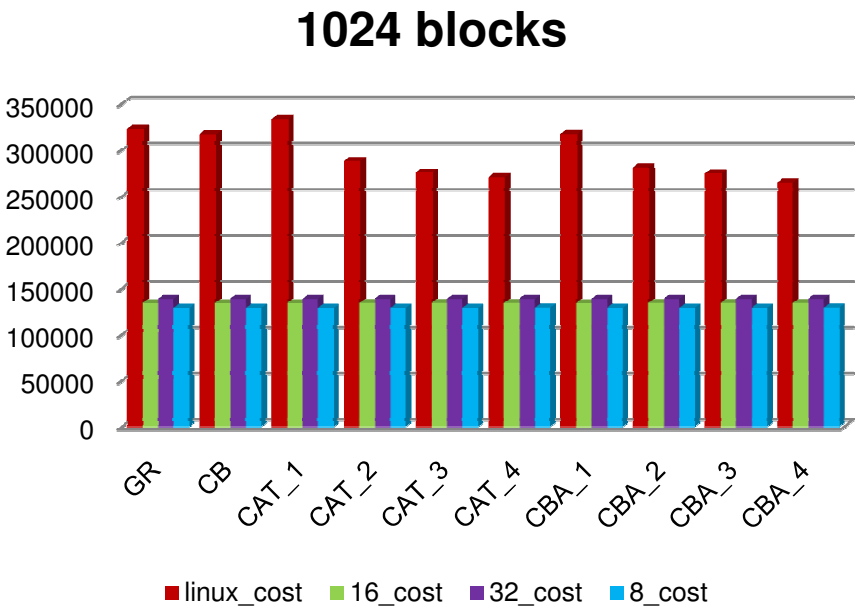




성능평가 결과

- GC Cost comparison

Cost = read_count + copy_count*10 + erase_count*75



method	cost
Linux swap (8)	260,000 ~ 330,000
LOBI (8)	128,000
LOBI (16)	133,000
LOBI (32)	137,000

→ 약 200% 의 성능 향상

결론

- **Workload 분석**
  - ✓ 리눅스 커널을 수정하여 스왑 트레이스를 수집
  - ✓ 스왑 트레이스 분석 및 스왑 시스템 시뮬레이션 프로그램 (swap-sim) 개발
  - ✓ 스왑 트레이스에서 스왑 아웃 및 스왑 인의 특성 분석

- **새로운 스왑 시스템 연구**
  - ✓ 로그 구조 방식의 스왑 슬롯 할당
  - ✓ 삭제 블록에 기반한 선반입(read-ahead) 스왑 인
  - ✓ 가비지 수집 성능 비교
  - ✓ 트레이스 기반 시뮬레이션을 통한 성능 검증