

LOBI: A swap scheme for flash memory storages

명지대학교 류연승

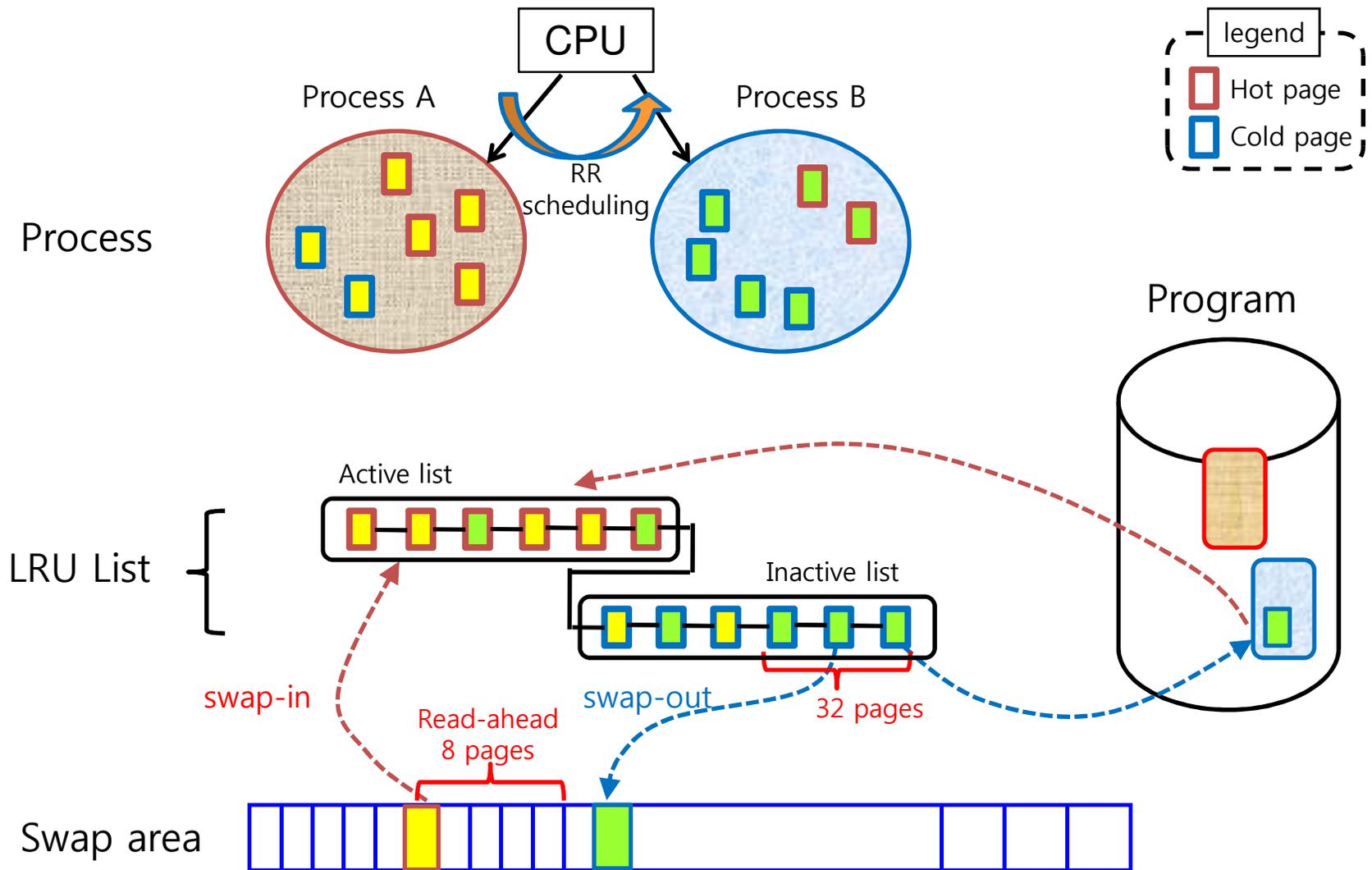
2008. 4. 25

NVRAMOS '08

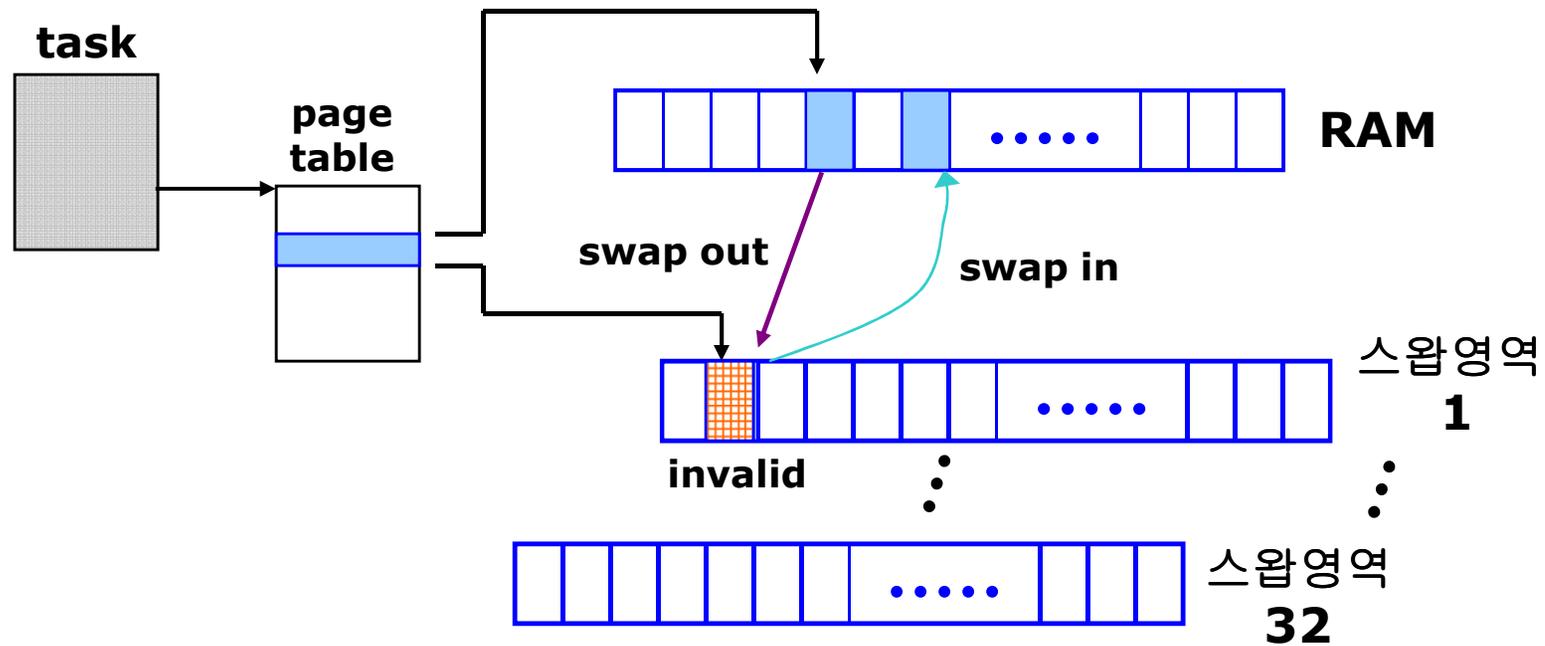
- 연구 목적과 연구 내용
- workload 분석
- 새 스왑 시스템
- 성능 평가
- 결론

- 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



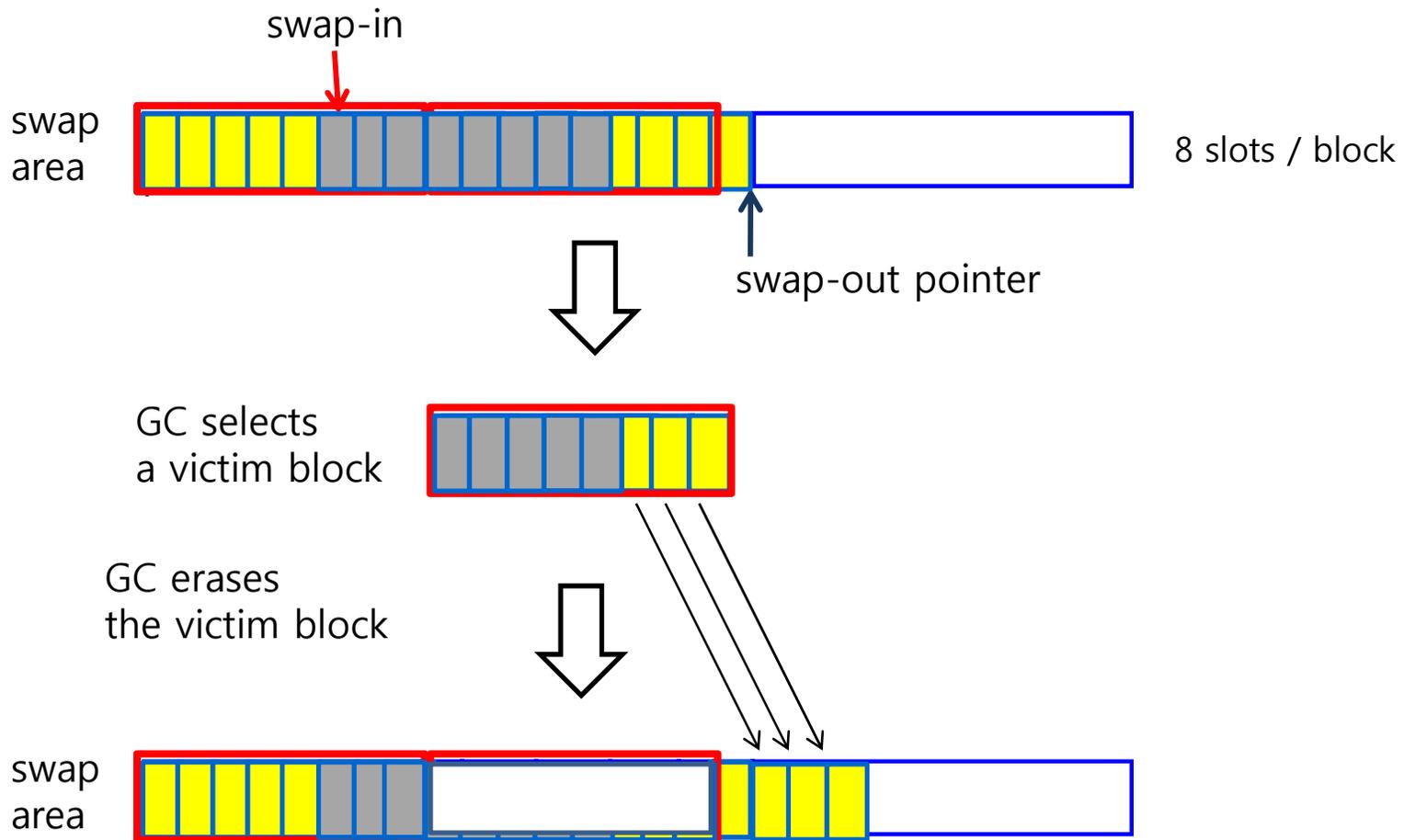
주요 특징

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

- 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



- GC cost
 - ✓ Victim 블록 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

- 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

```

#0 28009 b2e23000 3374 115 6 5 0 44 1 45 2008 01 25 09 46 40 76988033
#0 28009 b2e81000 3374 115 6 5 0 43 1 44 2008 01 25 09 46 40 76988041
#0 28009 96e2000 3374 115 6 3 0 42 1 43 2008 01 25 09 46 40 76988049
#0 28009 96f1000 3374 115 6 3 0 41 1 42 2008 01 25 09 46 40 76988057
#0 28009 b2e80000 3374 115 6 5 0 40 1 41 2008 01 25 09 46 40 76988065
#0 28009 b2e7f000 3374 115 6 5 0 39 1 40 2008 01 25 09 46 40 76988073
#0 28009 b2e7e000 3374 115 6 5 0 38 1 39 2008 01 25 09 46 40 76988081
#0 28009 b2e7c000 3374 115 6 5 0 37 1 38 2008 01 25 09 46 40 76988089
#0 28009 96e8000 3374 115 6 3 0 36 1 37 2008 01 25 09 46 40 76988097
#0 28009 96e8000 3374 115 6 3 0 35 1 36 2008 01 25 09 46 40 76988105
#0 28009 82ee000 3374 115 6 3 0 34 1 35 2008 01 25 09 46 40 76988113
#0 28009 82ec000 3374 115 6 3 0 33 1 34 2008 01 25 09 46 40 76988121
#0 28009 82eb000 3374 115 6 3 0 32 1 33 2008 01 25 09 46 40 76988129
#0 28009 82ea000 3374 115 6 3 0 31 1 32 2008 01 25 09 46 40 76988137
#0 80028009 b7ee000 1749 116 7 5 0 80 1 81 2008 01 25 09 46 40 76988505
#0 80028009 b7ee000 1749 116 7 5 0 79 1 80 2008 01 25 09 46 40 76988513
#0 80028009 b7ee000 1749 116 7 5 0 78 1 79 2008 01 25 09 46 40 76988521
#0 80028009 b7ee000 1749 116 7 5 0 77 1 78 2008 01 25 09 46 40 76988529
#0 80028009 b7ee000 1749 116 7 5 0 76 1 77 2008 01 25 09 46 40 76988537
#0 80028009 b7ee000 1749 116 7 5 0 75 1 76 2008 01 25 09 46 40 76988545
#0 80028009 b7ee000 1749 116 7 5 0 74 1 75 2008 01 25 09 46 40 76988553
#0 80028009 b7ee000 1749 116 7 5 0 73 1 74 2008 01 25 09 46 40 76988561
#0 80028009 b7ee000 1749 116 7 5 0 72 1 73 2008 01 25 09 46 40 76988569
#0 80028009 b7ee000 1749 116 7 5 0 71 1 72 2008 01 25 09 46 40 76988577
#0 80028009 b7ee000 1749 116 7 5 0 70 1 71 2008 01 25 09 46 40 76988585
#0 80028009 b7ee000 1749 116 7 5 0 69 1 70 2008 01 25 09 46 40 76988593
#0 80028009 b7ee000 1749 116 7 5 0 68 1 69 2008 01 25 09 46 40 76988601
#0 80028009 b7edf000 1749 116 7 5 0 67 1 68 2008 01 25 09 46 40 76988609
#0 80028009 80106000 1749 116 7 5 0 102 1 103 2008 01 25 09 46 40 76988617
#0 80028009 80105000 1749 116 7 5 0 101 1 102 2008 01 25 09 46 40 76988625
#0 80028009 80104000 1749 116 7 5 0 100 1 101 2008 01 25 09 46 40 76988633
#0 80028009 80103000 1749 116 7 5 0 99 1 100 2008 01 25 09 46 40 76988641
#0 80028009 80102000 1749 116 7 5 0 98 1 99 2008 01 25 09 46 40 76988649
#0 80028009 80101000 1749 116 7 5 0 97 1 98 2008 01 25 09 46 40 76988657
#0 80028009 80100000 1749 116 7 5 0 96 1 97 2008 01 25 09 46 40 76988665
#0 80028009 800ff000 1749 116 7 5 0 95 1 96 2008 01 25 09 46 40 76988673
#0 80028009 800fe000 1749 116 7 5 0 94 1 95 2008 01 25 09 46 40 76988681
#0 80028009 800fd000 1749 116 7 5 0 93 1 94 2008 01 25 09 46 40 76988689
#0 80028009 800fc000 1749 116 7 5 0 92 1 93 2008 01 25 09 46 40 76988697
#0 80028009 800fb000 1749 116 7 5 0 91 1 92 2008 01 25 09 46 40 76988705
#0 80028009 800fa000 1749 116 7 5 0 90 1 91 2008 01 25 09 46 40 76988713
#0 80028009 800f9000 1749 116 7 5 0 89 1 90 2008 01 25 09 46 40 76988721
#0 80028009 800f8000 1749 116 7 5 0 88 1 89 2008 01 25 09 46 40 76988729
#0 80028009 800f7000 1749 116 7 5 0 87 1 88 2008 01 25 09 46 40 76988737
#0 80028009 800f6000 1749 116 7 5 0 86 1 87 2008 01 25 09 46 40 76988745
#0 80028009 800f5000 1749 116 7 5 0 85 1 86 2008 01 25 09 46 40 76988753

```

- 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

- 트레이스 개요

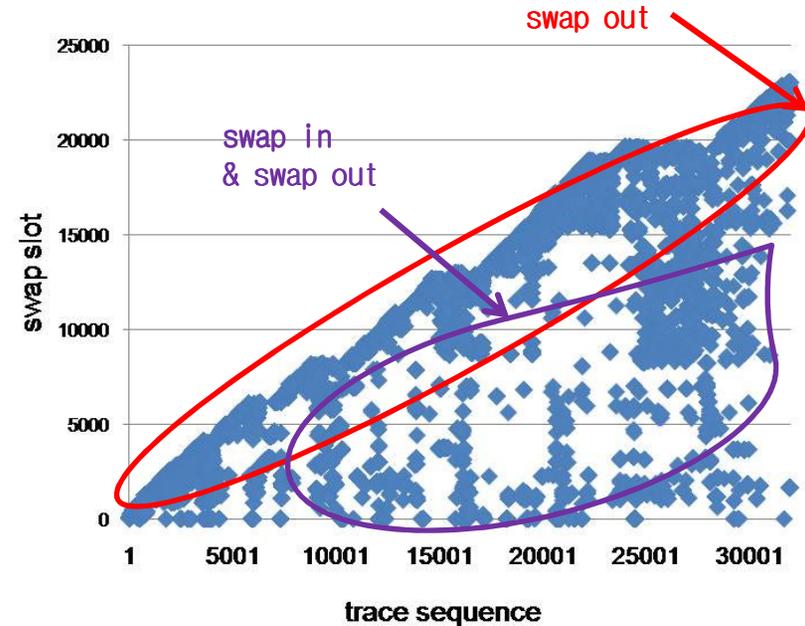
 - ✓ 스왑 개수

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

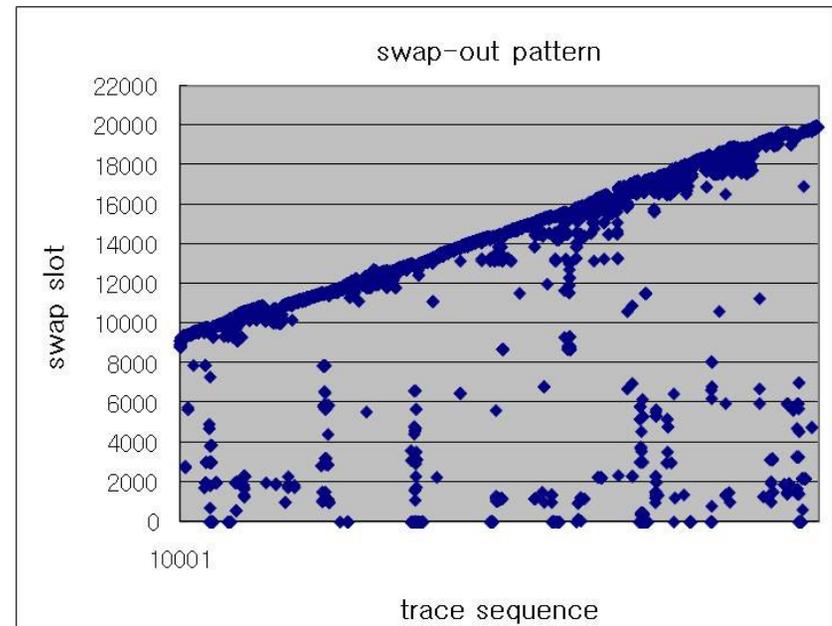
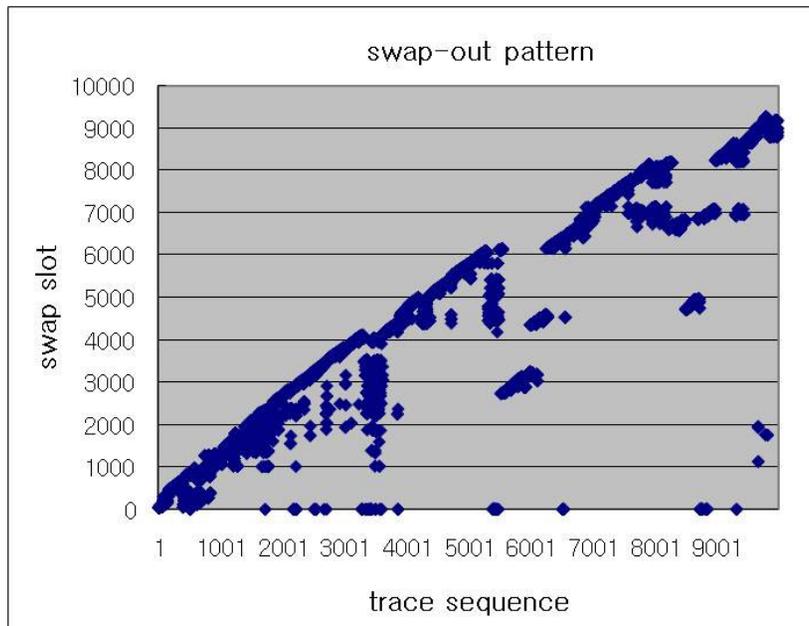
 - ✓ 프로세스 개수

	count
Process	127

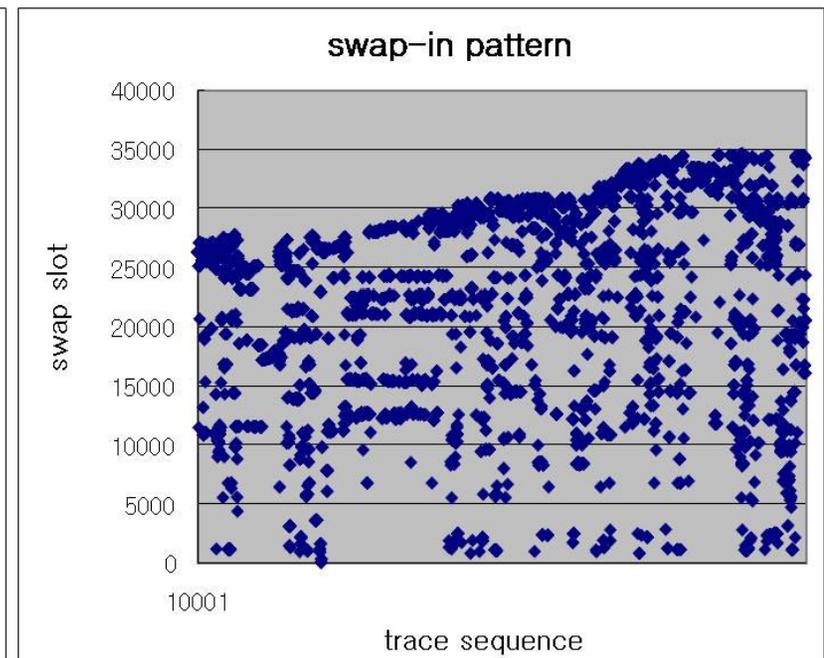
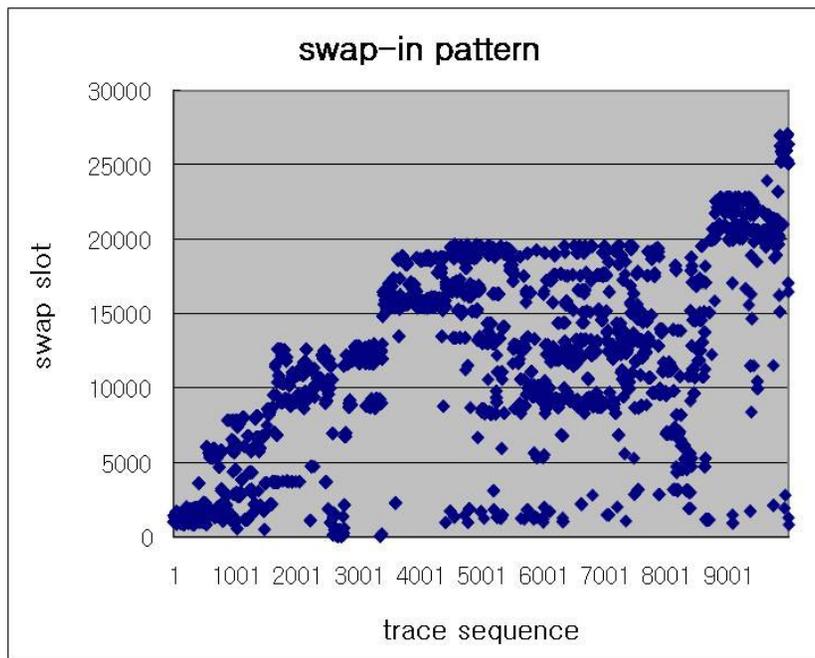
- 스왑이 시작되면 스왑 슬롯을 순차적으로 할당
- page fault에 의해 스왑 인이 발생
- 점차 스왑 영역의 전반에 걸쳐 스왑 아웃과 스왑 인이 발생



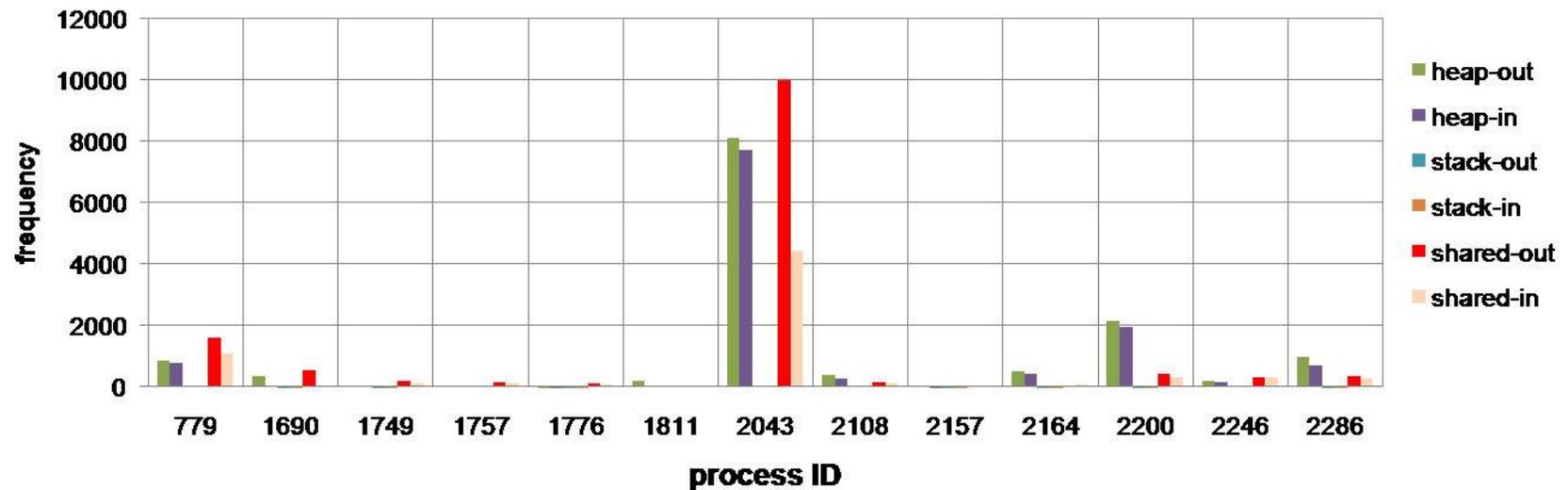
- 스왑 아웃
 - 순차적으로 슬롯을 할당하여 사용하다가, 스왑인되어 비는 슬롯들도 사용하게 된다.



- 스왑 인
 - 스왑 영역의 전반에 발생하고 있다.

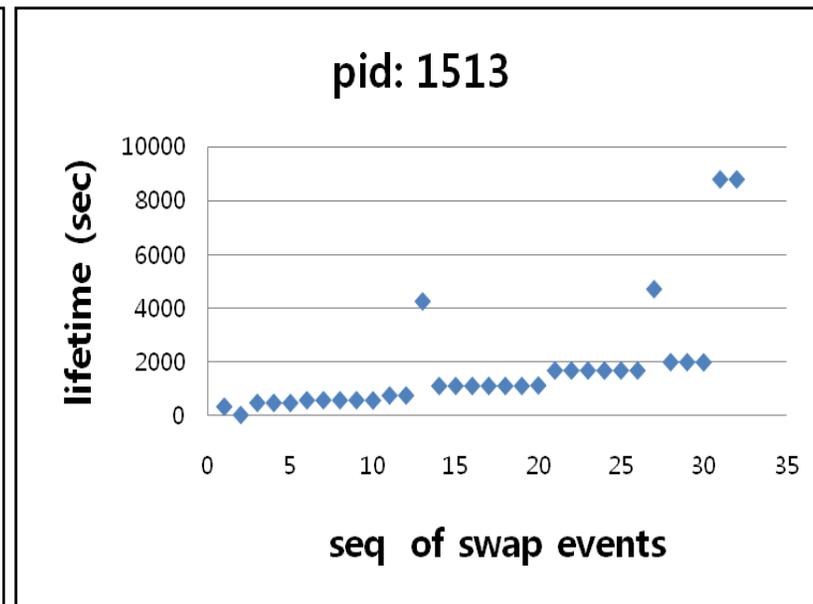
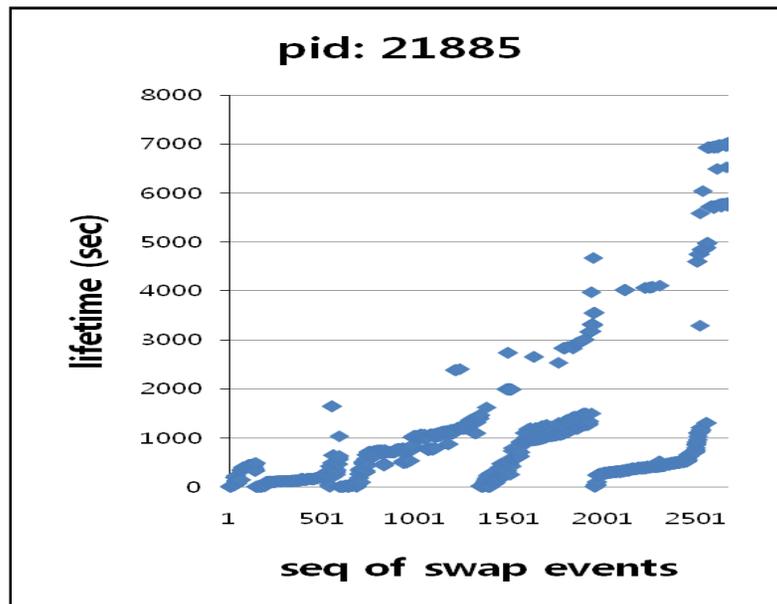


- 프로세스 분석
 - ✓ 특정 소수의 프로세스들이 많이 스왑된다.

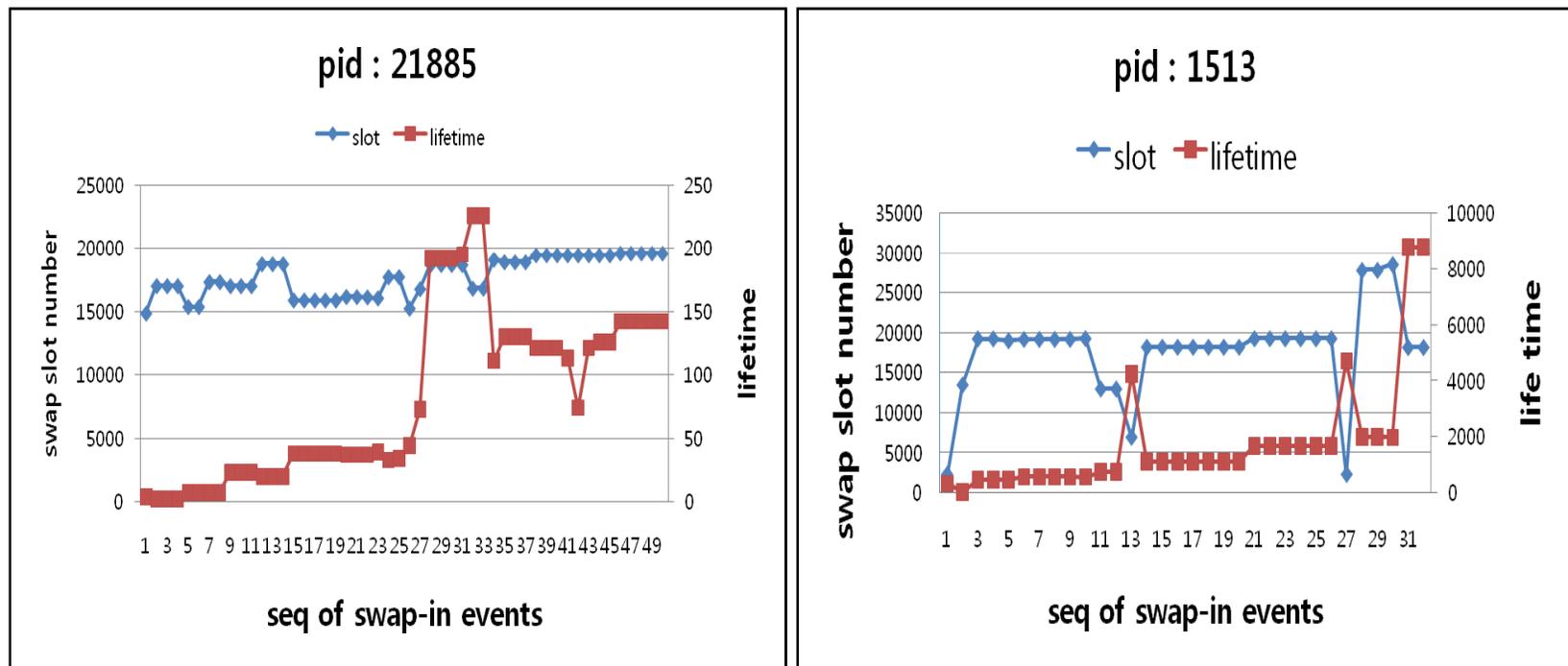


스왑 트레이스 분석 : lifetime

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



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



▪ 고려사항

스왑 워크로드 특성

• 동일한 프로세스의 동일 유형 페이지들이 비슷한 시기에 그룹으로 스왑 아웃됨

• 프로세스별로 스왑 횟수의 차이가 크다

• 비슷한 시기에 스왑 인되어 유사한 lifetime을 가짐

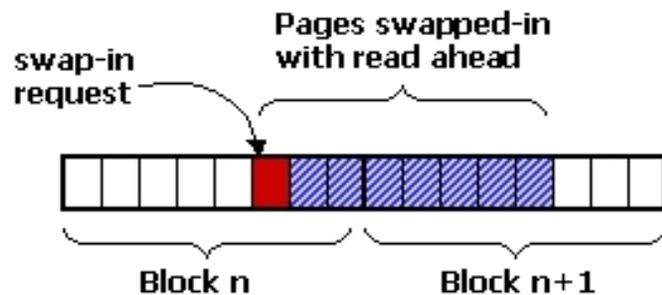
• 프로세스별로 따로 저장

• 스왑 아웃 페이지들을 순차적으로 저장

• 여러 개 페이지를 선반입

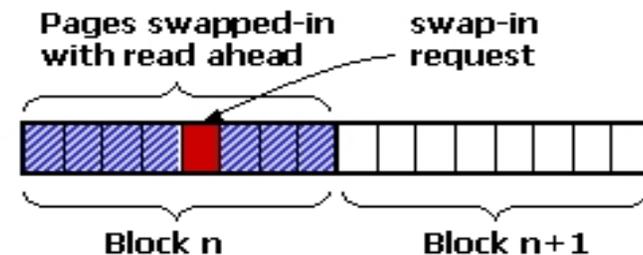
LOBI (Log-structured swap-Out, Block-aligned swap-In)

- 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

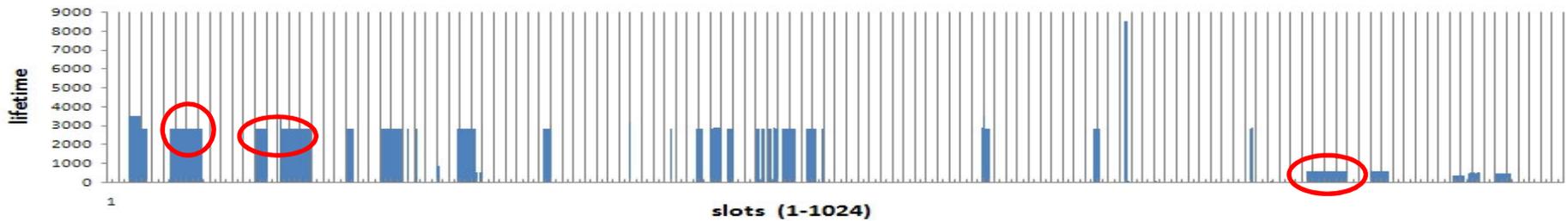


< 삭제 블록 기반 선반입 >

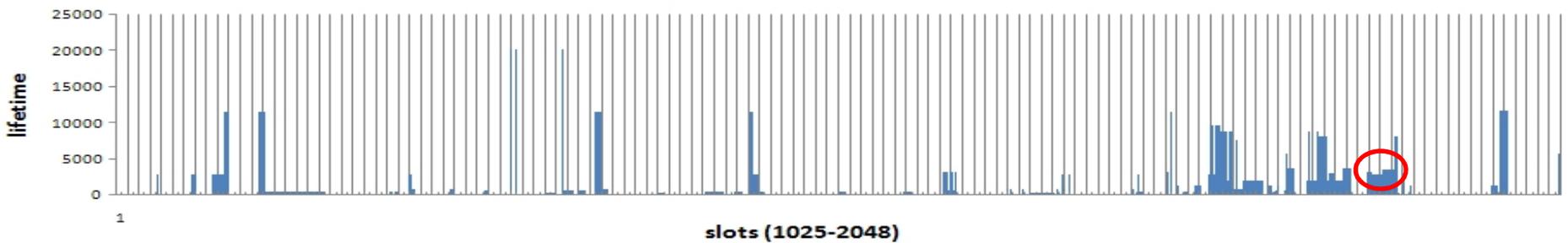
한 블록 내에서만 선반입을 수행

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

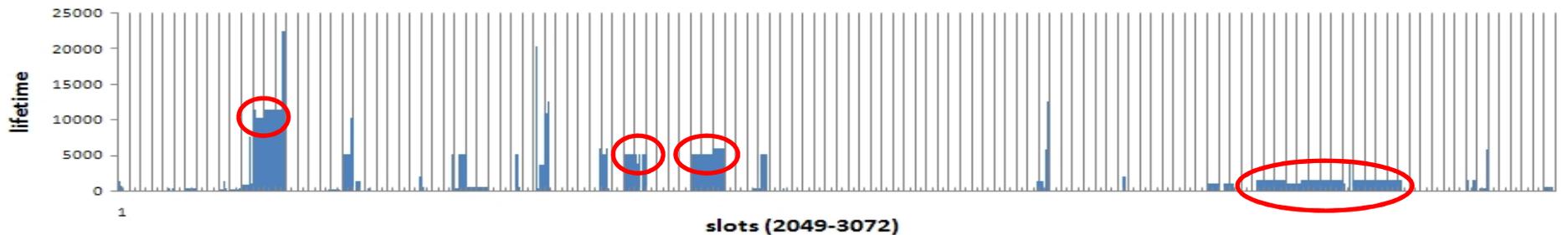
lifetime vs. 8 slots



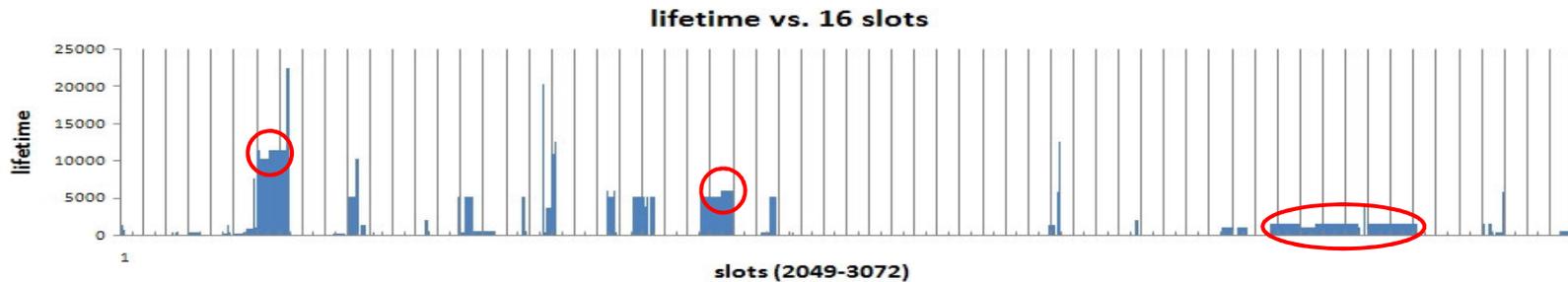
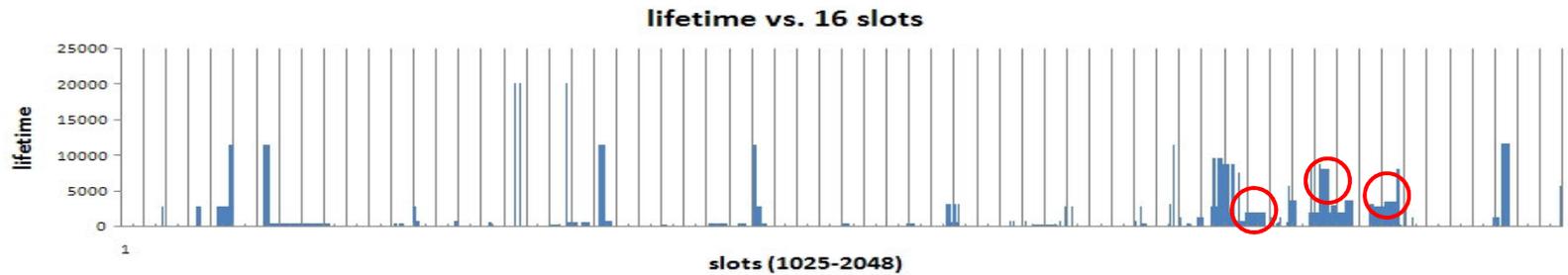
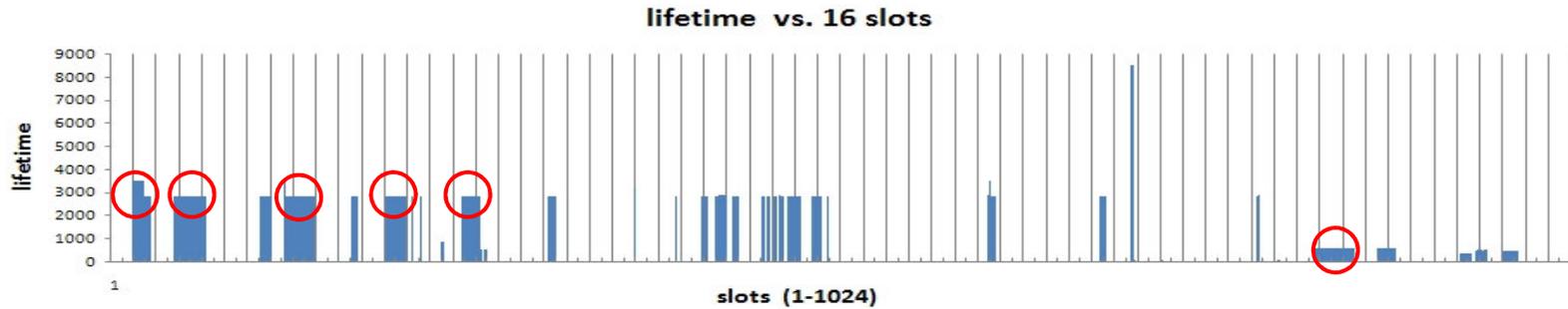
lifetime vs. 8 slots



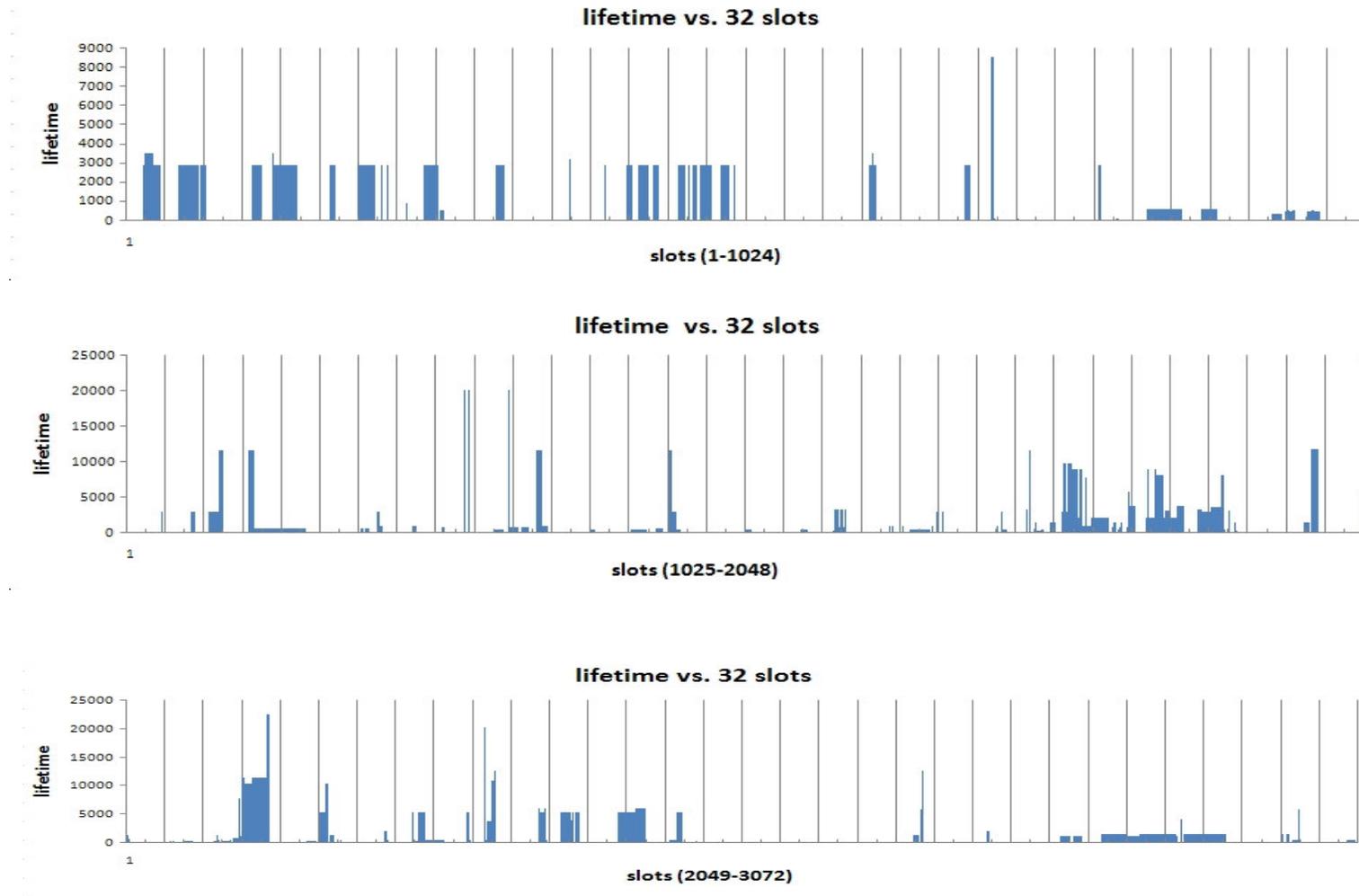
lifetime vs. 8 slots



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



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



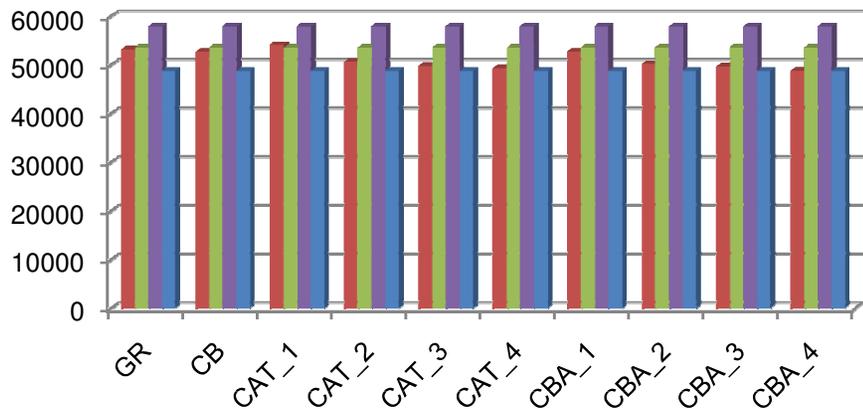
- Trace-driven simulation
- Linux swap scheme
- LOBI
 - ✓ Block-aligned 8 pages
 - ✓ Block-aligned 16 pages
 - ✓ Block-aligned 32 pages
- Garbage collection algorithm
 - ✓ Greedy (GR)
 - ✓ Cost-Benefit (CB)
 - ✓ Cost-Age-Time with Age (CATA) : victim blocks 1~4
 - ✓ Cost Benefit with Age (CBA) : victim blocks 1~4

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

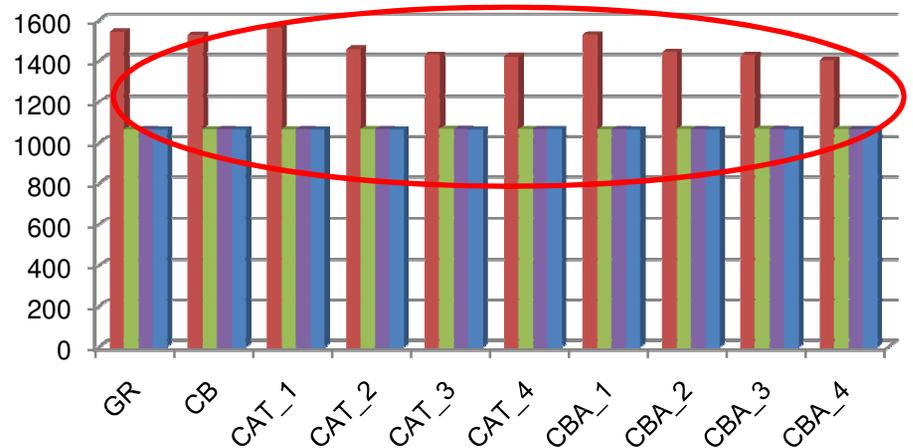
- Performance metric
 - ✓ read count
 - ✓ erase count
 - ✓ erase copy count
 - ✓ wear-leveling degree
 - ✓ garbage collection cost

성능평가 결과

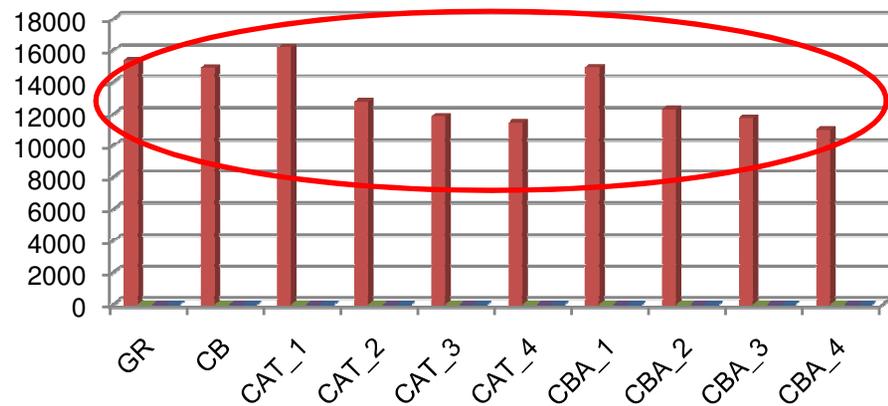
- 1024 blocks



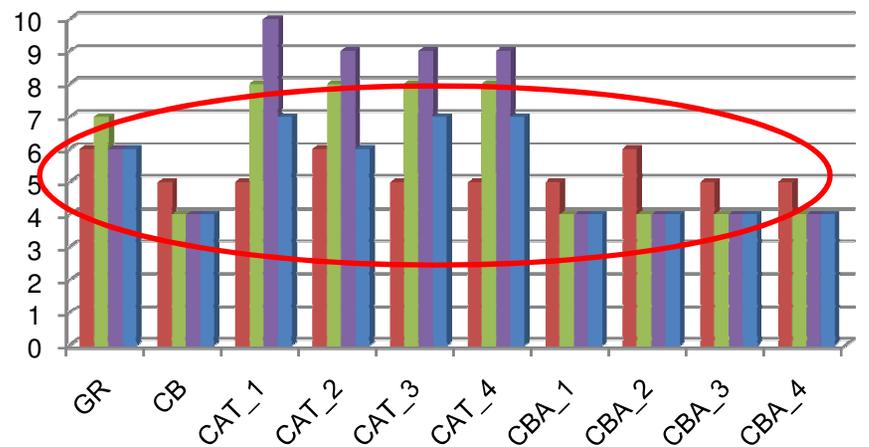
■ linux_read_count ■ 16_read_count ■ 32_read_count ■ 8_read_count



■ linux_erase_count ■ 16_erase_count ■ 32_erase_count ■ 8_erase_count

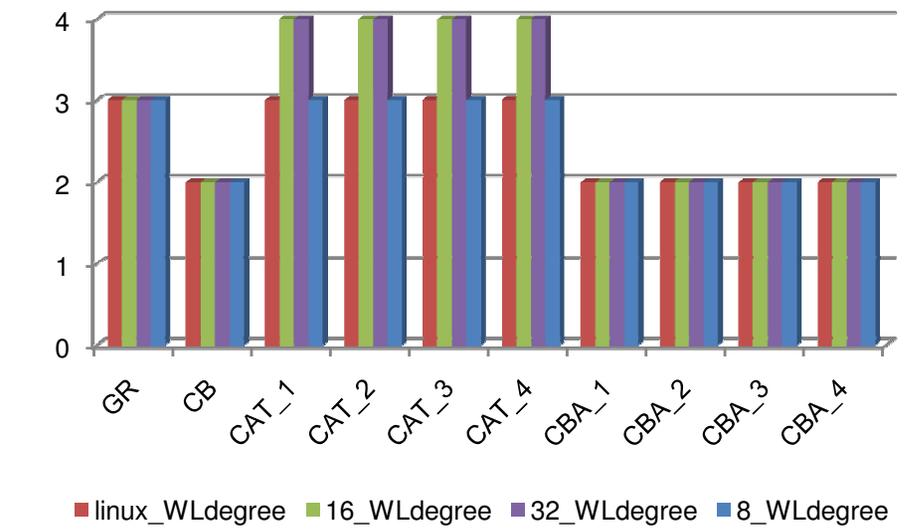
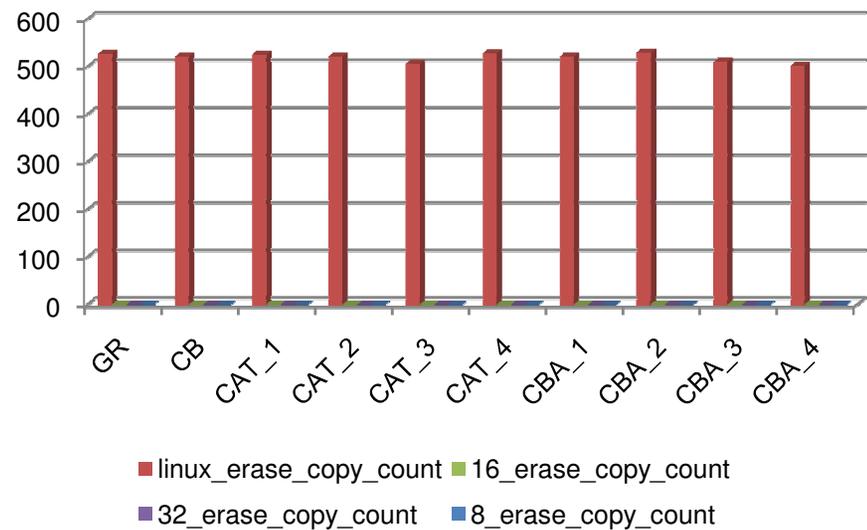
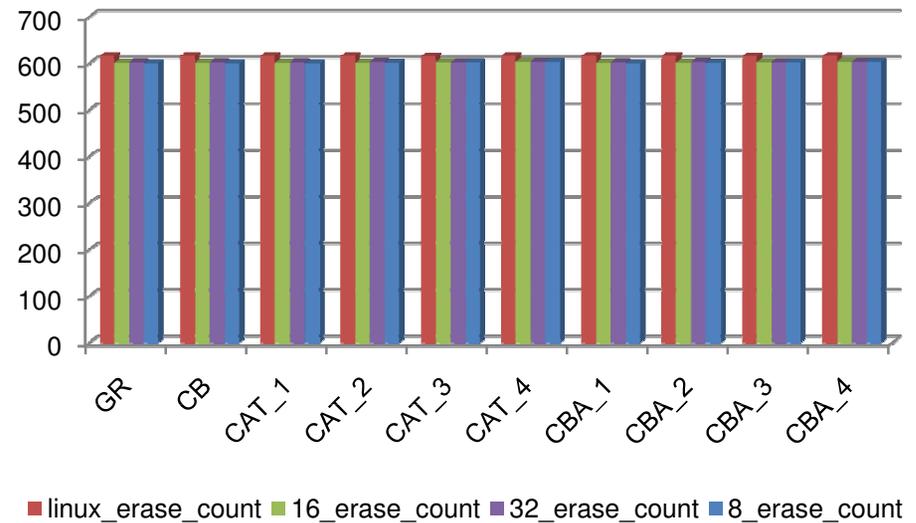
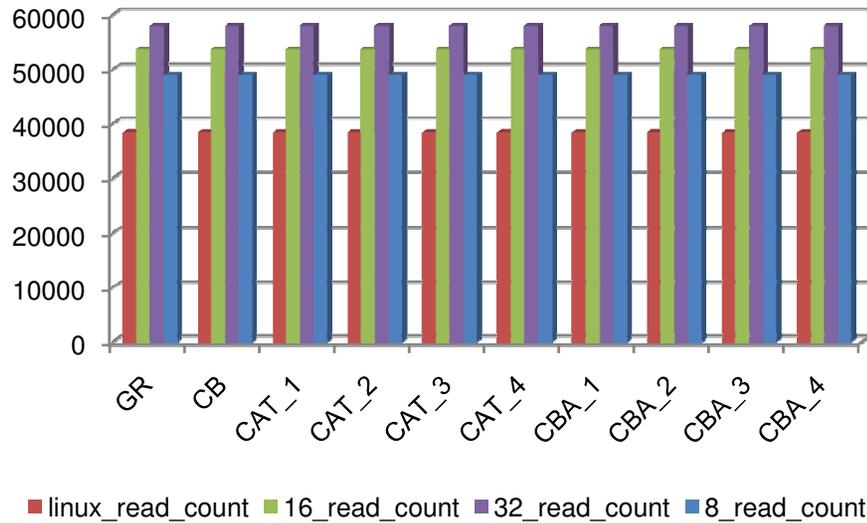


■ linux_erase_copy_count ■ 16_erase_copy_count
■ 32_erase_copy_count ■ 8_erase_copy_count



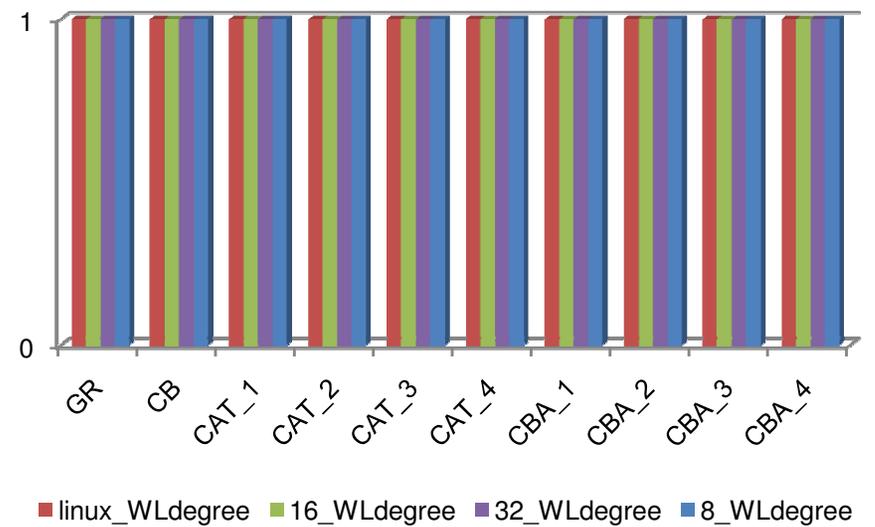
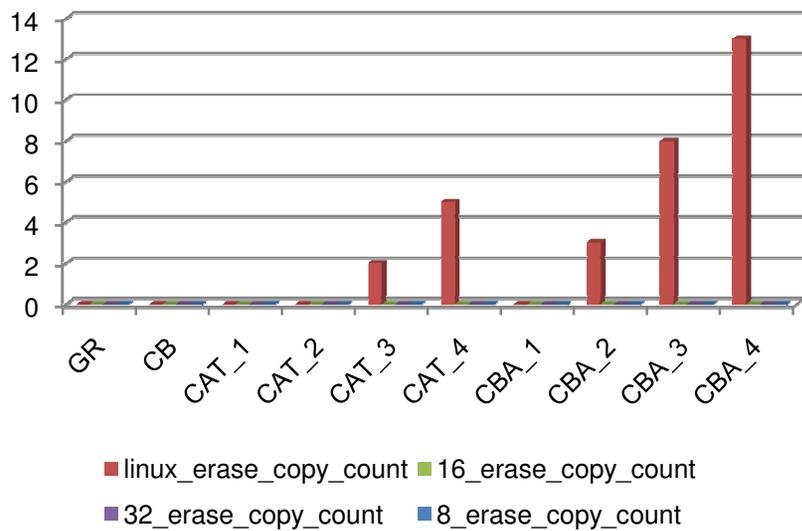
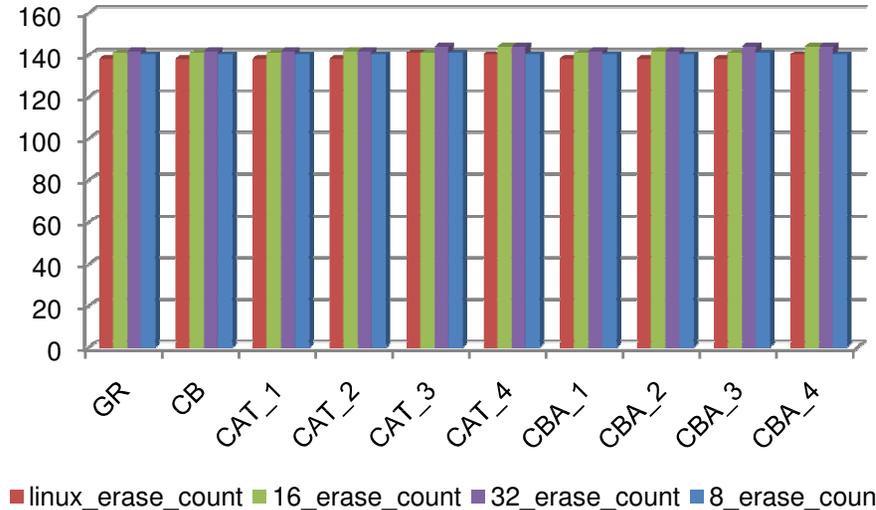
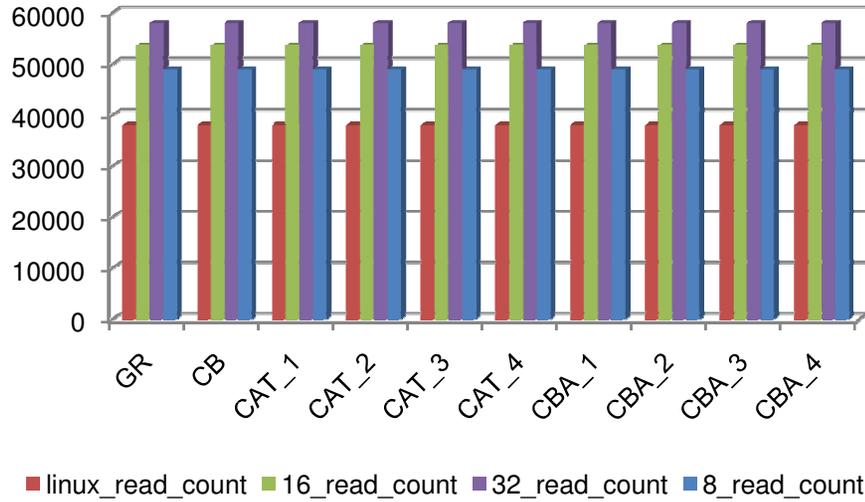
■ linux_WLdegree ■ 16_WLdegree ■ 32_WLdegree ■ 8_WLdegree

1536 blocks



성능평가 결과

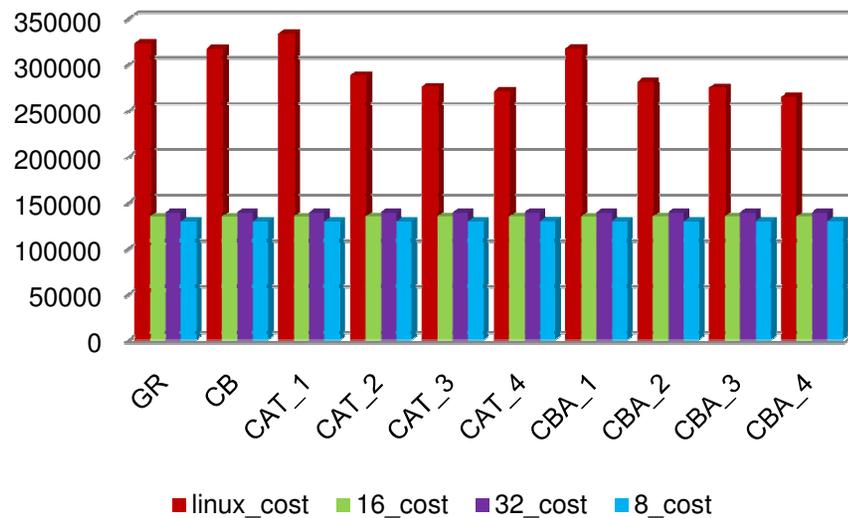
2048 blocks



- GC Cost comparison

$$\text{Cost} = \text{read_count} + \text{copy_count} * 10 + \text{erase_count} * 75$$

1024 blocks



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

▪ 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