The Design of a Flash-based Linux Swap System

Yeonseung Ryu
Myongji University
October, 2008

Contents

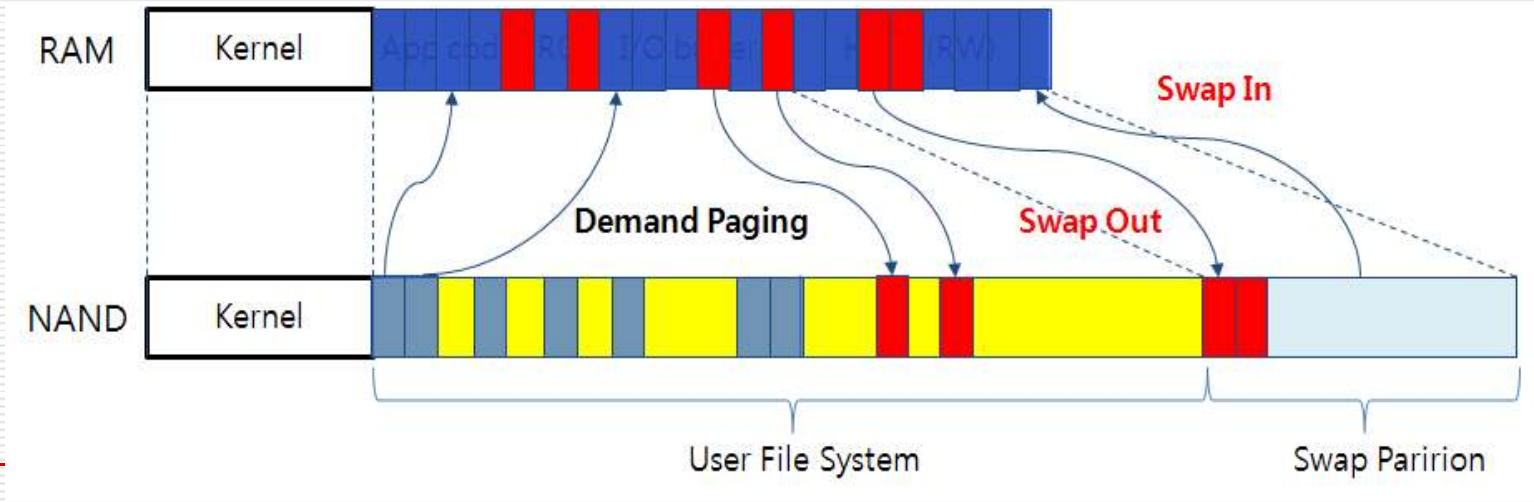
- **Overview of linux Swap System**
  - How does the swap system operates?
  - What are the problems of flash based swap system?
- **New Swap System**
  - Segment-based swap space management and fast start-up
  - Block-aligned read-ahead swap-in scheme
  - Performance evaluation

Introduction

- All modern general-purpose OS use virtual memory techniques. Most of virtual memory implementations divide the virtual address space of an application program into pages; Page size is usually 4KB in linux.

- When an application executes, its contents (code and data) are loaded on demand into page frames of main memory.

- The stack and heap of the process are created in the main memory on demand.



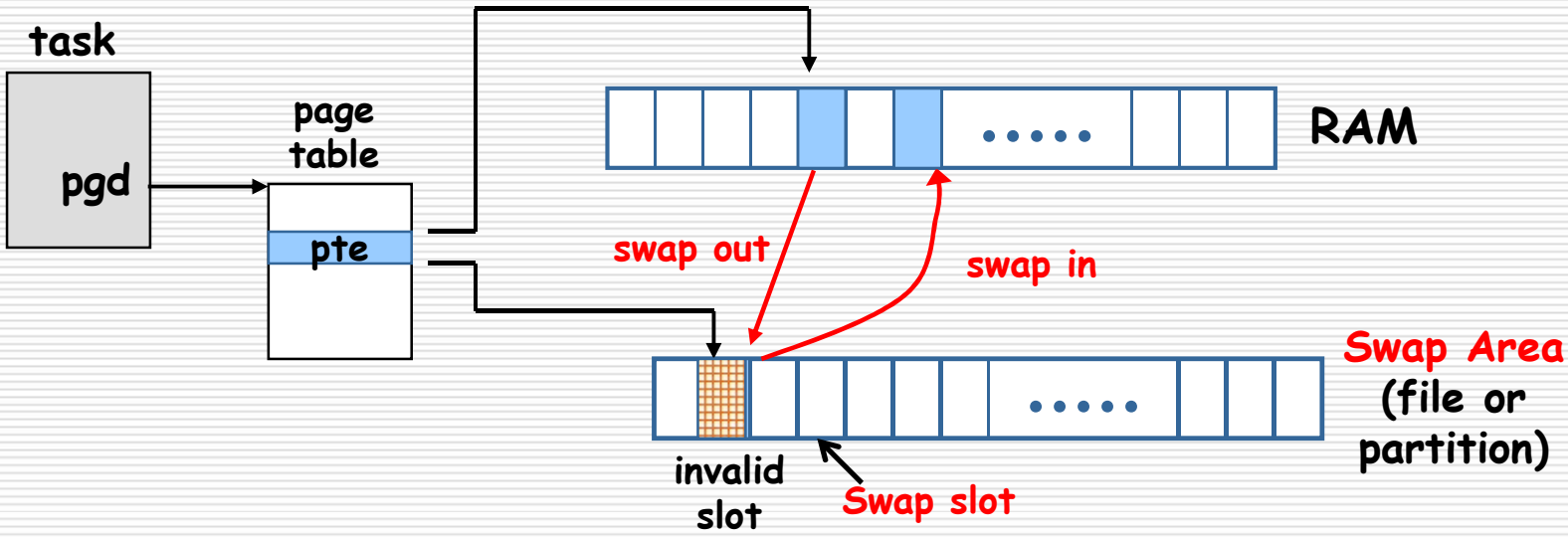
Introduction

- When the free memory is almost used up, the Linux kernel performs page frame reclaiming. Page frame reclaiming procedure picks up victim page frames and makes them free.

- If a victim page frame is mapped to a portion of a disk file and the page is dirty, the kernel writes the content of the page frame to the corresponding disk file.

- If a victim page frame is not mapped to a disk file like stack and heap (it is called as anonymous pages), the kernel saves the page contents in a dedicated disk partition (or a disk file) called swap area.

Linux Swap System



- A swap area consists of a sequence of swap slots: 4KB blocks used to contain a swapped-out page.
- When an anonymous page is selected for page reclamation, it is swapped out to the swap area.
- Swap-in operation occurs when a process attempts to address a page that has been swapped out.
- When a page is swapped-in from swap area to main memory, we call its slot invalid slot.

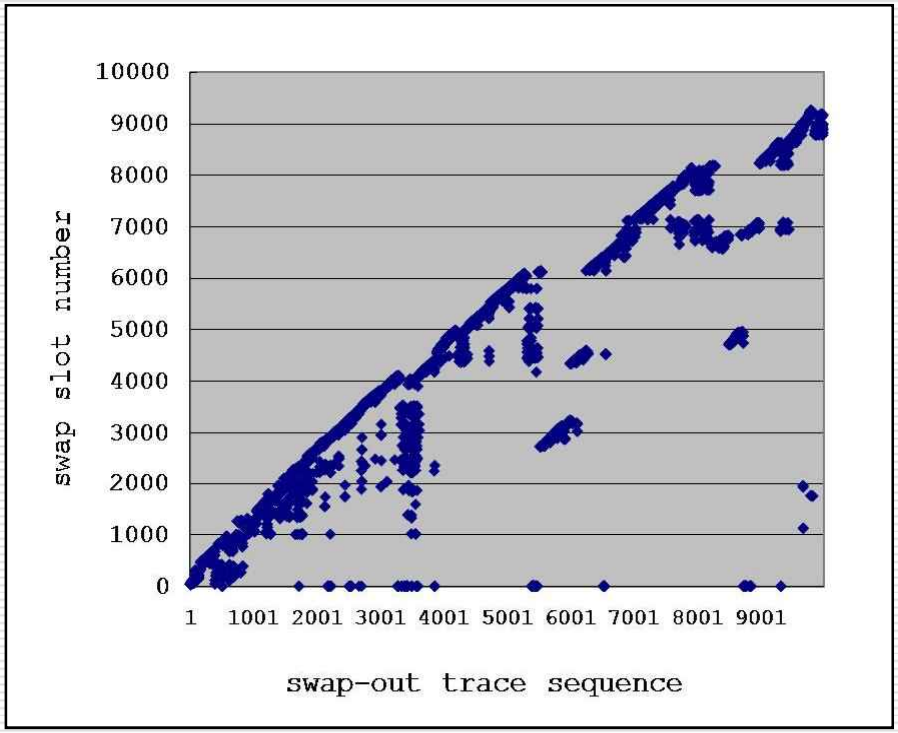
Swap-out

- Linux swap system has been optimized to reduce disk seek time.
- In order to minimize disk seek time, the kernel tries to store swap-out pages in contiguous slots and thus allocates slots from the last allocated slot.
- This simple approach, however, can increase the average seek time during swap-in operations because many occupied swap slots may be scattered far away from one another.

Swap-out

- In order to address this problem linux kernel restarts allocation from the beginning of the swap area whenever 256 free slots were allocated after the last restart from the beginning of the swap area.

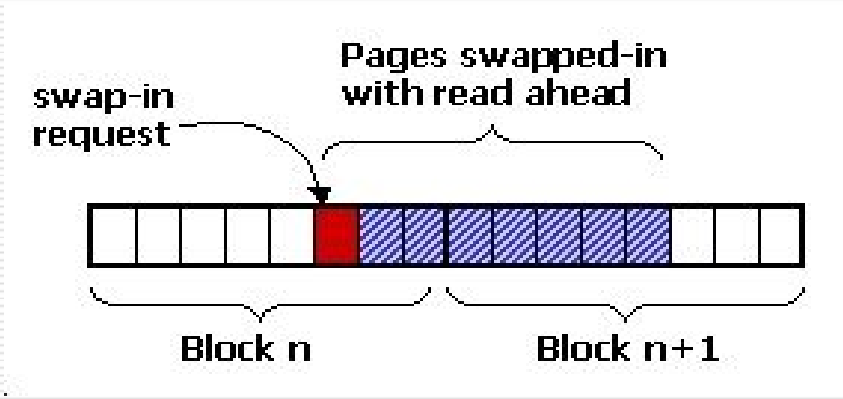
- So, kernel uses slots again that became free due to swap-in requests.



- However, when flash memory is used as swap device, reusing these free slots requires erase operations.
- Moreover, since flash memory does not require seek operation, kernel does not need to restart allocation from the beginning of the swap area.

Swap-in

- A swap-in operation tries to read contiguous eight pages including the requested one. Why read-ahead ?
  - Locality property : neighbor pages tend to be accessed soon.
  - Seek time: some consecutive pages are read at a time.
- When flash memory is used as swap device, however, swap-in operation with read-ahead can invalidate the swap slots that lie over two erase blocks.
- In order to reuse these eight slots to store swapped-out pages, we need two block erase operations.



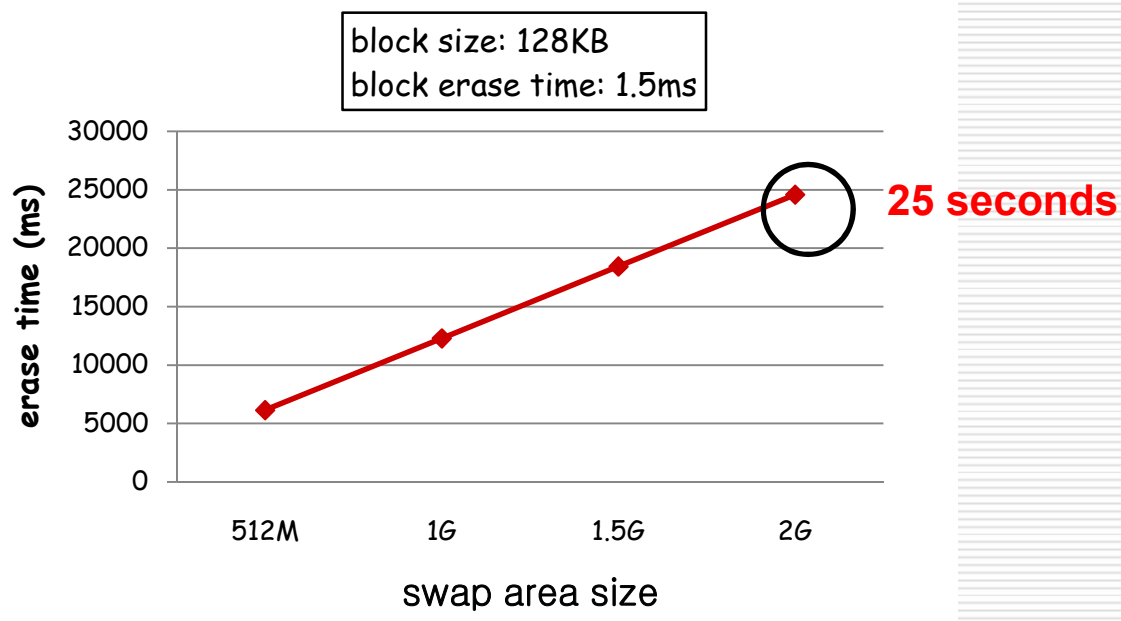
Linux swap-in

New swap system

- **Considerations**
  - Fast start-up
  - Minimizing garbage collection cost
  - Wear-leveling

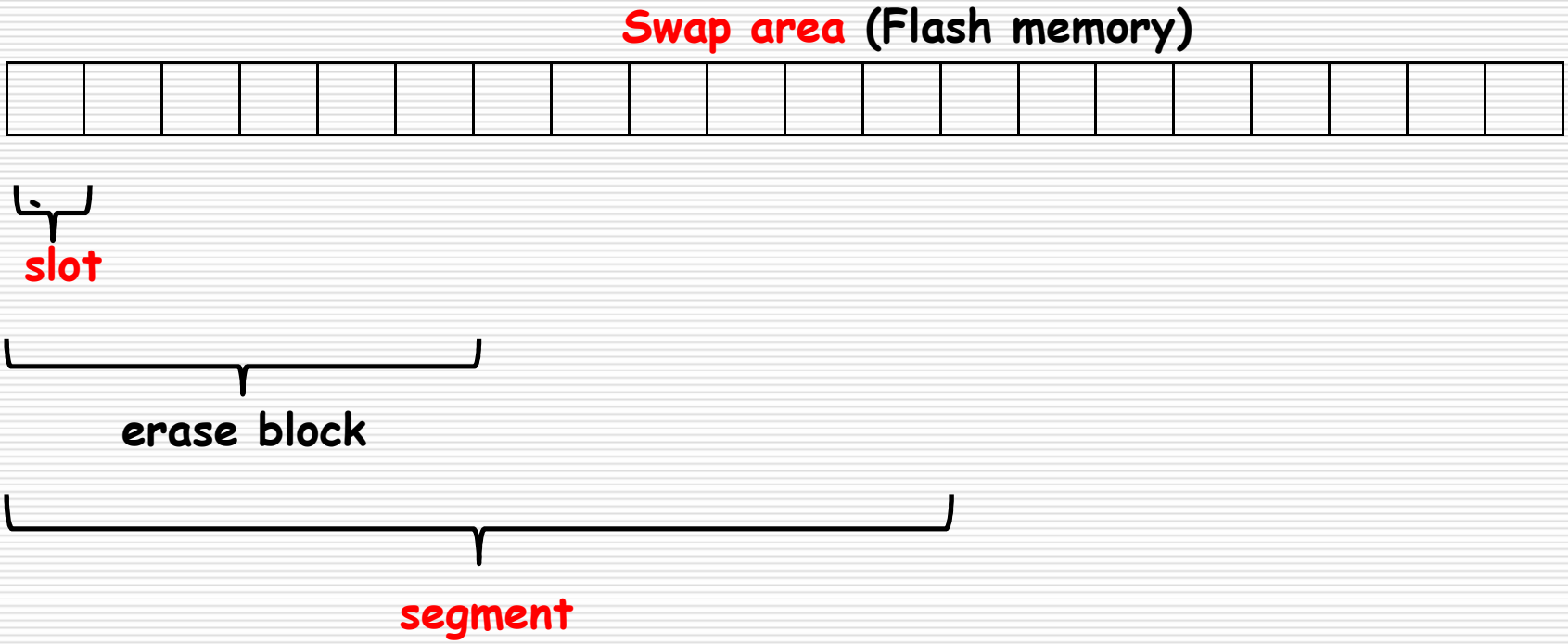
fast start-up ?

- When the system is booted, kernel needs to clean entire swap space since the previous data in the swap area are obsolete.
- This cleaning operation makes system start-up time longer..
- Worst cleaning time vs. swap area size



Swap space management

- Patent: Segment based swap space management, October, 2008
  - Swap space is divided into segments.



Segment-based swap space management

- Each segment consists of a set of erase blocks.
- Each segment has a segment header which contains segment status (used/free) and erasure counter.

- A segment is an unit of erase. That is, all the blocks in a segment are erased together at a time.
- When the segment is erased, segment status becomes free and erasure counter is increased by 1.

Segment-based swap space management

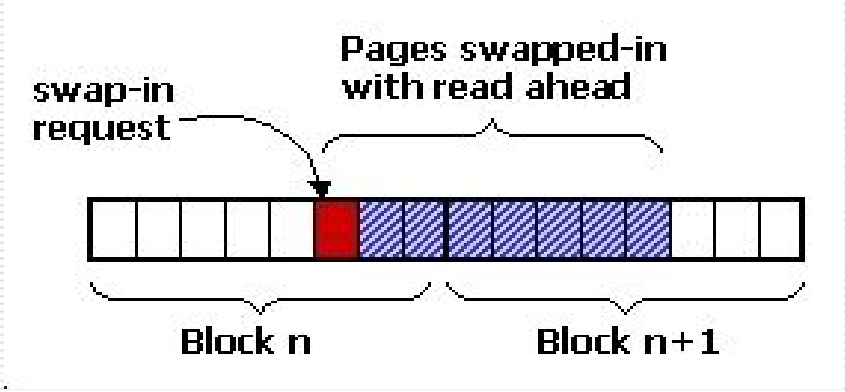
- During the start-up, kernel scans all segment headers and constructs data structure about segments in the RAM .
- After constructing segment data structure, kernel determines the number of segments which must be erased by start-up procedure.
  - If there is a given limit of start-up time, start-up procedure can clean only a few segments.
  - Length of start-up time vs. amount of free swap space
- If the kernel does not clean all invalid segments, remaining invalid segments will be erased by garbage collection process afterwards.
- This space management scheme can limit start-up time.

Segment-based swap space management

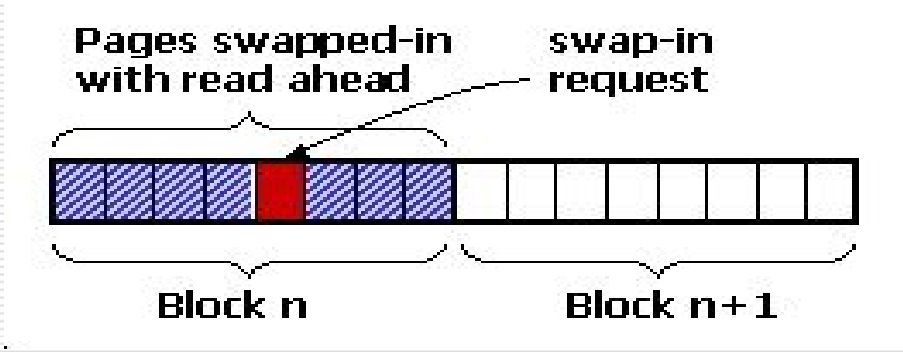
- We are studying ...
  - Segment size
  - Start-up time limit
  - How many segments are erased by start-up procedure
  - Garbage collection algorithm
  - Wear-leveling
  - etc

Block-aligned read-ahead swap-in

- Patent: Read-ahead swap-in method considering flash memory erasure block, May. 2008



Linux swap-in



Block-aligned swap-in

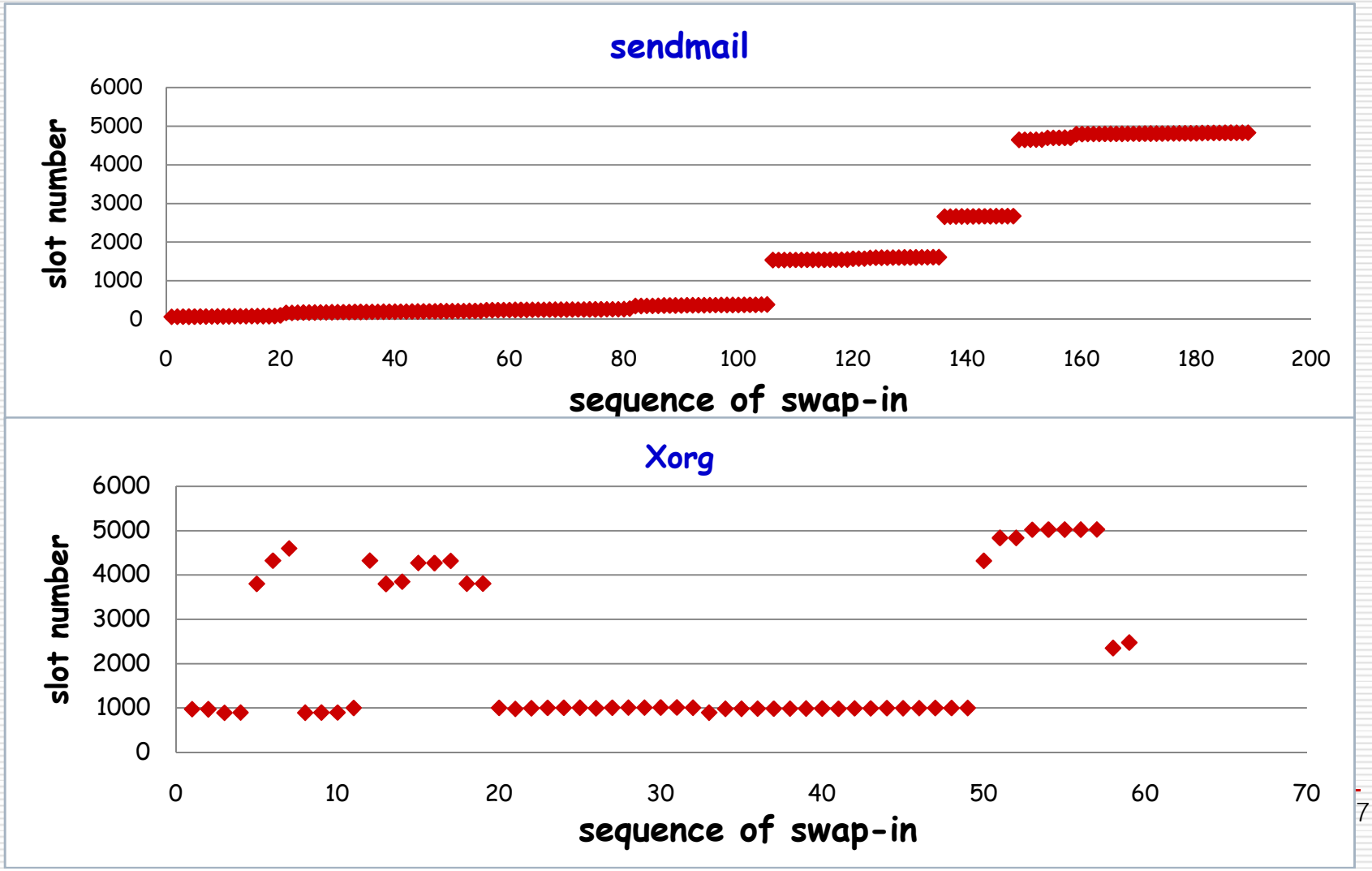
- Proposed scheme reads ahead pages that lie in the same block.
  - We need only one erase operation to reuse swapped-in slots.
- By reducing the number of invalid blocks to be erased, we can decrease the garbage collection (GC) cost.

Analysis of Linux swap I/O traces

- In order to evaluate performance, we collected some swap I/O traces from linux kernel.
- Most of all, we want to know the swap-in behaviour. So, we turned off read-ahead option of swap-in before collecting the traces.
- Because we disabled read-ahead option, all swapped-in pages were loaded on demand and we can examine the pure swap-in behaviour.
- We examined if the locality exists in the swap slot accesses due to swap-in operations.
- We found that the temporal/spatial locality exist in the swap-in references

Locality of Swap-in pattern

- These figures show the slot numbers accessed by swap-in operations of a particular process.



Read ahead ??

- Due to locality, read-ahead is a good approach to reduce the disk seek time.

- Flash memory does not require seek time but read-ahead can result in better performance.
  - Read-ahead can decrease the number of page faults

Performance Evaluation

- We have performed trace-driven simulation to investigate the performance of read-ahead swap-in schemes and garbage collection algorithms.

- Swap-out : allocates slots sequentially

- Swap-in: read-ahead
  - 8 pages by linux swap scheme
  - 8 pages by block-aligned scheme
  - 16 pages by block-aligned scheme
  - 32 pages by block-aligned scheme

- Garbage collection algorithms
  - Greedy (GR)
  - Cost-Benefit (CB)
  - Cost-Age-Time (CAT)
  - Cost Benefit with Age (CBA)

Performance Evaluation

- **Garbage collection algorithms**
  - Greedy (GR) : selects a block with the largest amount of invalid slots
  - Cost-Benefit (CB) : selects a block that maximize the formular:

$$\frac{age \cdot (1-u)}{1+u}$$

age **: the time since the most recent modification**
u **: the fraction of space occupied by valid slots**

  - Cost-Age-Time (CAT) : selects a block that maximize the formular:

$$\frac{1-u}{1+u} \cdot age \cdot \frac{1}{erase\_count}$$

  - Cost Benefit with Age (CBA) : selects a block like CB and sorts valid slots with ages and moves the oldest pages first

Performance Evaluation

- Through the simulation, we measured the following performance metrics to calculate the garbage collection cost.
  - read count
  - erase copy count : the number of writes due to GC
  - erase count

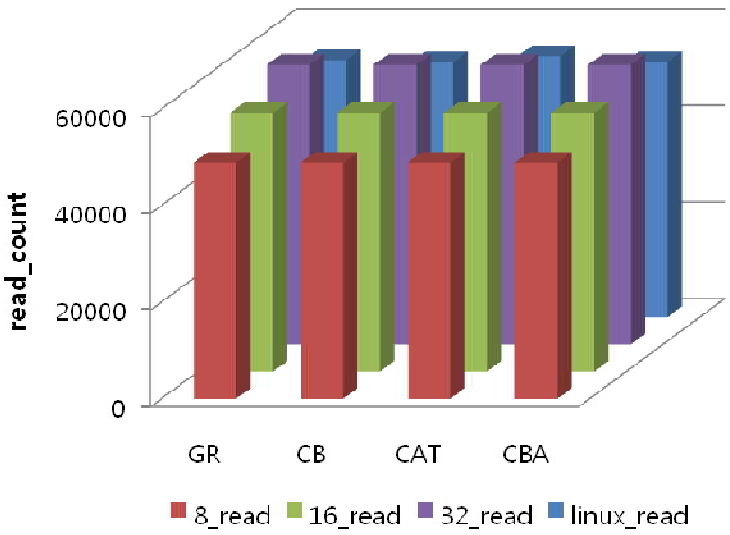
- Garbage collection must copy the valid slots in the victim block to the free space before erasing it.
- Copy operation requires read and write operation.

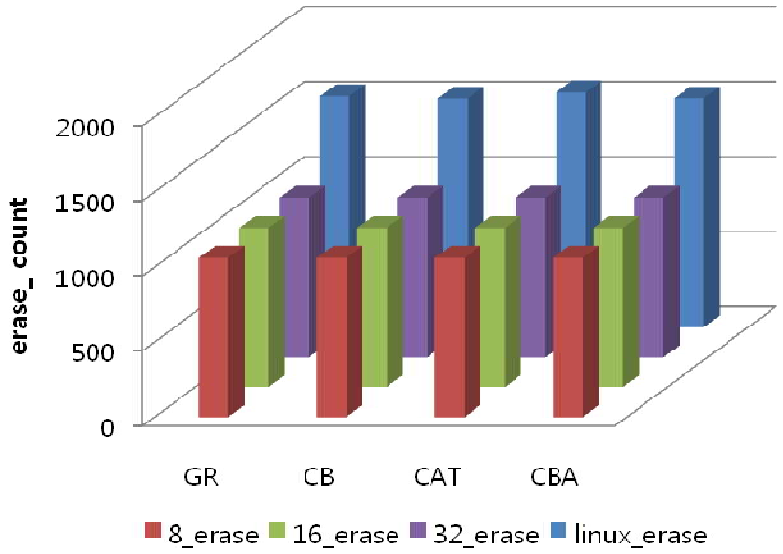
- Garbage collection cost

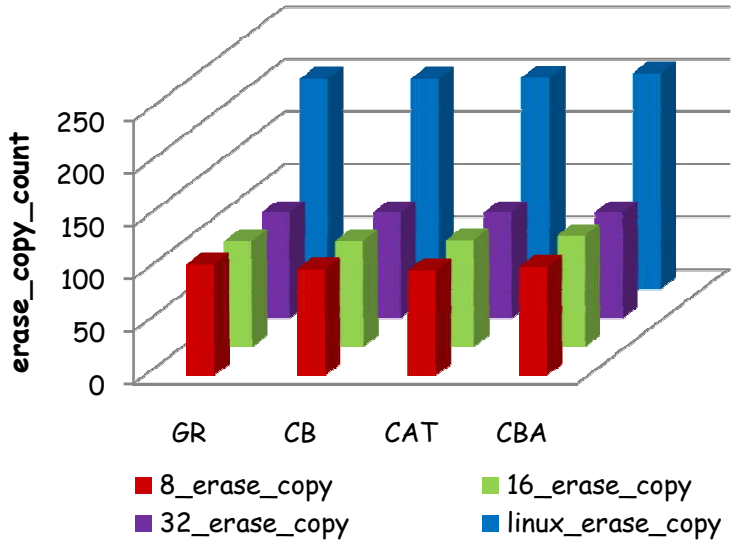
Cost = read_count + copy_count*10 + erase_count*75

  - the write operation is 10 times slower than the read operation and the erase operation is 75 times slower than the read operation.

Some Results



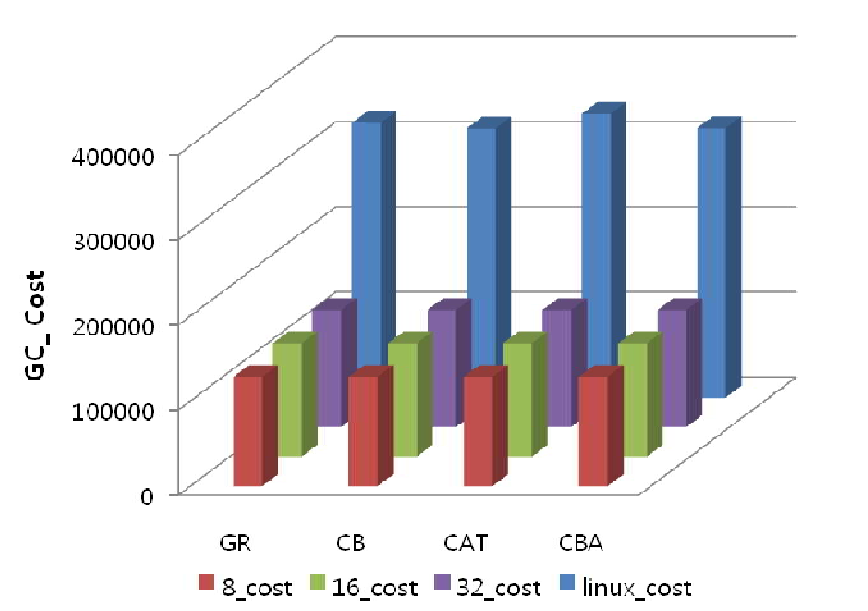




- GC algorithms have little effect.
- The number of read-ahead has little effect.
- Block-aligned read-ahead outperforms non-block-aligned read-ahead.

Some Results

- GC Cost



- As a result, GC cost of proposed block-aligned swap-in scheme is almost three times smaller than linux scheme.

Conclusions

- Segment based swap space management can reduce the start-up time.
- Block-aligned read-ahead scheme can reduce the garbage collection cost.
- We are yet studying ..

Thank you..

ysryu@mju.ac.kr


Swap-in
request