CSCI315/ECEN427: Swap Space Simulation

2025-SPRNG

Dr. Ahmed Elnokrashy

Shorouk Sherif
Amenah Medhat
Ranim Hisham
Hana Hatem
Khetam Mohammed
Noor Alzoghby

What is swap space?

Swap space is a portion of the hard drive used as an extension of RAM when physical memory is full.
It allows the system to run more applications .

xv6 initially lacks a swap space implementation.

and so

Our problem is to find a way to implement swap space in xv6 to improve its ability to handle memory-intensive tasks.

Implementation Overview

- Allocate a designated region on the disk to function as the swap area.
- Modify the xv6 kernel to handle page faults triggered when a process accesses a page not currently in RAM.
- Implement page replacement algorithms to decide which pages to swap out when memory is full.
- Update the page table management to reflect pages swapped in and out

Approaches: Page Replacement Algorithms

FIFO (First-In, First-Out)

- Pages are swapped out in the order they were brought into memory.
- Easy to implement but often performs poorly.

LRU (Least Recently Used)

- Swaps out the page that has not been used for the longest time.
- Difficult and expensive to implement perfectly

Clock cycle

- Maintains a circular list of pages in memory.
- Each page has a "reference bit." When a page is accessed, indicating recent use.

LFU (Least Frequently Used)

- Swaps out the page that has been used least frequently.
- Can suffer from the "new page problem"

Implementation Details: Clock Cycle approach

Data Structures:

- A circular linked list of pages in memory.
- A reference bit for each page.
- A clock hand that points to the "current" frame being inspected

Algorithm Steps:

When a page needs to be swapped out:

1. On a page fault, start from the current position of the clock hand.
2. Traverse pages in a circular list until a victim is found:

If the reference bit is 1 :

- Reset it to 0 (give a second chance).
- Move clock hand forward.

If the reference bit is 0 :

- Swap out the selected page and load the new page into its place.
- Set the new page's reference bit to 1 and advance the clock hand.

Results

```
kalloc: Memory full! Attempting swap out...
select_victim: Found victim VA 0x40001000 in proc 2, PTE 0x8000AAAA # (Example victim from select_victim)
SWAPOUT: Swapping out VA 0x40001000 PA 0x8000BEEF to swap slot 0 # (Your swapout print)
swap_write_page: Writing page from PA 0x8000BEEF to disk block 1000 # (Your swap_write_page print)
kalloc: Memory full! Attempting swap out...
select_victim: Found victim VA 0x40002000 in proc 2, PTE 0x8000BBBB
SWAPOUT: Swapping out VA 0x40002000 PA 0x8000CAFE to swap slot 1
swap_write_page: Writing page from PA 0x8000CAFE to disk block 1001

usertrap: Page fault at VA 0x40001000, scause 13 # (Page fault for a load access to a swapped page)
usertrap: Detected PTE_SWAPPED for VA 0x40001000. Calling swapin.
swapin: Swapping in VA 0x40001000 from swap slot 0 to new PA 0x8000DEAD # (Your swapin print)
swap_read_page: Reading page to PA 0x8000DEAD from disk block 1000 # (Your swap_read_page print)

usertrap: Page fault at VA 0x40002000, scause 15 # (Page fault for a store access to a swapped page)
usertrap: Detected PTE_SWAPPED for VA 0x40002000. Calling swapin.
swapin: Swapping in VA 0x40002000 from swap slot 1 to new PA 0x8000F00D
swap_read_page: Reading page to PA 0x8000F00D from disk block 1001

swaptest: Allocated and accessed 200MB of memory.
swaptest: Data integrity verified. All swapped pages read back correctly.
swaptest: SUCCESS!
$
```

Challanges:

1. **Performance Overhead**: Swapping is slow because accessing swap space on disk is much slower than accessing RAM.
2. **Synchronization**: Ensuring shared data is protected from race conditions during concurrent access.
3. **Kernel Complexity:** Implementing swap space increases the complexity of the kernel's memory management.
4. Choosing the Right Replacement Algorithm

Thank you