COL331: Operating Systems
Assignment 3 Report

Sanyam Garg — 2022CS11078
Aneeket Yadav — 2022CS11116

April 30, 2025

1 Introduction

This report describes the swapping mechanism implemented in xv6 to enable execution of processes that require more memory than the available physical RAM. When memory is low, pages are swapped out to disk and brought back in on demand. This allows better memory utilization and system responsiveness under memory pressure.

2 Swap Partition Design

We extend the xv6 disk layout by adding a dedicated swap area between the `sb block` and the `log` of the disk layout. The swap space is divided into 800 **swap slots**, where each slot consists of 8 disk blocks (4096 bytes), the size of one page. These slots are managed using an array of metadata structures:

- `pageperm (int)`: Stores the original page permissions.
- `isfree (int)`: A flag indicating whether the slot is currently free (1) or occupied (0).

The swap slots are initialized during boot and stored separately from the file system structures to avoid interference with normal file operations.

3 Memory Management Enhancements

The physical memory limit is enforced by setting:

```
#define PHYSTOP 0x400000
```

The system is modified to handle out-of-memory situations as follows:

- The function `swapout()` is called to free space.
- Each process has an `rss` field (resident set size) that counts how many user pages it currently holds in RAM. This field is updated whenever a new page is allocated and helps in deciding the victim page.

4 Page Replacement and Swapping Out

To swap out pages, the kernel iterates through all processes and selects a victim page using the following policy:

- Choose the process with the maximum `rss` value.
- Within that process, select the first user page that is present (PTE_P) and not recently accessed (PTE_A = 0).

Once a victim page is selected:

1. Find a free swap slot.
2. Write the page content to the slot on disk.
3. Update the PTE to store the slot index and clear the PTE_P bit. Set a custom flag PTE_PG to mark the page as swapped.
4. Free the physical memory and update `rss`.

5 Page Fault and Swapping In

If a process accesses a swapped-out page, a page fault (T_PGFLT) is triggered. The page fault handler does the following:

1. Read the faulting address from the `cr2` register.
2. Extract the swap slot index from the PTE.
3. Allocate a new physical page.
4. Read the page content from disk into memory.
5. Restore the original PTE permissions and mark the page as present.

6 Impact of α and β

In virtual memory systems, swapping is a crucial mechanism that allows the operating system to handle memory pressure by moving pages between RAM and disk. The efficiency of this process depends on the **adaptive page replacement strategy**, governed by two key parameters:

- α (**ALPHA**): Controls how aggressively the system increases the number of pages swapped out
- β (**BETA**): Determines how quickly the system reduces the free-page threshold

This report analyzes how different values of α and β influence:

1. Swapping frequency
2. Memory utilization
3. System responsiveness

7 Mathematical Formulation

The adaptive strategy updates two variables dynamically:

7.1 Threshold Update

$$Th_{new} = \left\lfloor Th_{current} \times \left(1 - \frac{\beta}{100}\right) \right\rfloor \tag{1}$$

Where:

- Higher $\beta \rightarrow$ Faster threshold reduction $\rightarrow$ More frequent swapping
- Lower $\beta \rightarrow$ More conservative memory reclamation

7.2 Page Count Update

$$Npg_{new} = \min\left(\text{LIMIT}, \left\lfloor Npg_{current} \times \left(1 + \frac{\alpha}{100}\right) \right\rfloor\right) \tag{2}$$

Where:

- Higher $\alpha \rightarrow$ More pages swapped per operation
- Lower $\alpha \rightarrow$ Gradual increase in swap volume

8 Behavior Analysis

Table 1: Impact of Different Parameter Combinations

Case	**Parameters**	**Advantages**	**Disadvantages**
Aggressive	$\alpha = 100$, $\beta = 10$	Fast memory recovery	Risk of over-swapping
Conservative	$\alpha = 25$, $\beta = 10$	Stable performance	Slow response to pressure
Balanced	$\alpha = 50$, $\beta = 25$	Good compromise	Requires tuning

9 Conclusion

The analysis demonstrates that:

- α controls swap batch size
- β determines sensitivity to memory pressure
- Optimal values are workload-dependent

Recommended configurations:

- **Interactive systems**: $\alpha = 40$, $\beta = 20$
- **Batch processing**: $\alpha = 80$, $\beta = 30$
- **General purpose**: $\alpha = 60$, $\beta = 25$