COL331/633 Project Report: Copy-On-Write in xv6

Your Name

April 24, 2024

1 Introduction

This project aims to implement the Copy-On-Write (COW) mechanism in the xv6 operating system. COW is a resource management technique enabling parent and child processes to initially share memory pages without copying them. When a process writes to a shared page, a copy is made, ensuring data integrity. This report presents the design and implementation of COW in xv6.

2 Design

2.1 Enable Memory Page Sharing

To implement the Copy-On-Write (COW) mechanism, we introduce a reverse map (r-map) data structure called `page_to_refcnt`. This structure keeps track of the number of processes referencing the same memory page.

```
int page_to_refcnt[PHYSTOP >> PTXSHIFT];
```

- We utilize the `setupkvm()` function to create a new page directory.
- We iterate through each page table entry using `walkpgdir` and perform the following actions:
  - Obtain the flags and physical address of the page mapped to the page table entry.
  - Utilize the `mappages` function to map the same physical address to the page table entry in the new directory table, retaining the same permissions except for write access.
  - Increment the count for the number of processes mapped to this physical page.
  - Load the address of the new directory table into the CR3 register.

2.2 Handle Writes on Shared Pages

We introduce a `swapslothdr` struct to store required data for a swapped block on disk, including the count of mapped processes. Additionally, we have introduced a `swapslothdr` array to keep track of each swapped block. `page_to_refcnt`. This structure keeps track of the number of processes referencing the same memory page.

When a process writes to a shared page, a page fault occurs. We implemented a page fault handler that creates a new page with the same contents as the shared page. The process's page table entry is then updated to point to the new page.

- The CR2 register holds the virtual address of the page that caused the page fault.
- We retrieve the page table entry in the directory table and check if the present bit is set.
- If the present bit is set:
  - If the number of processes mapped to the page address is less than or equal to 1, we grant write access in the page table entry and load the directory table address into the CR3 register.
  - If the number of processes mapped to the page address is greater than 1, we allocate a new page in memory using the `kalloc()` function. We then copy the content from the original page to the new page in memory, map its physical address to the same page table entry, and decrement the count of the original page in the `page_to_refcnt` array.
- If the present bit is not set:
  - We allocate the new page using the `kalloc` function.
  - We write the content of the respective swap block on disk to the new page allocated in memory.
  - We assign the count in the `page_to_refcnt` array to the count stored in the `swap_slots` array.
  - We assign the physical address of the allocated page with permissions retrieved from `swap_slots` to the page table entry.

2.3 COW with Swapping