RV Educational Institutions ®
RV College of Engineering ®

Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi

Approved by AICTE, New Delhi

EXPERIENTIAL LEARNING 2023-24

IMPLEMENTATION OF COPY ON WRITE ON FORK IN XV6

Faculty Name : Dr Jyoti Shetty (DEPARTMENT OF COMPUTER SCIENCE ENGINEERING)

R.V. College of Engineering, Bangalore-59

ABSTRACT

This project explores the integration of Copy-On-Write (COW) functionality into the fork system call in XV6, aiming to optimize memory management and resource utilization. The implementation enhances system performance and scalability, empowering users to tailor XV6 to their unique requirements. Through meticulous exploration of kernel configuration options and system call modifications, the project elevates understanding of memory management techniques in operating system design.

INTRODUCTION

In operating systems, efficient memory management is paramount for optimizing system performance and resource utilization. However, traditional methods of memory allocation and process creation pose challenges in terms of overhead and redundancy.

During fork, a common issue is the need to duplicate the parent's memory space for the child, consuming both time and resources. Hence we use Copy-On-Write (COW), a technique devised to alleviate the inefficiencies of memory duplication during process creation.

XV6 RISC-V OPERATING SYSTEM

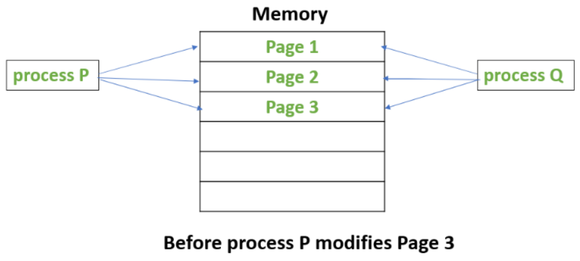
- Xv6 is a Unix-based operating system that provides students with a structured yet accessible environment to delve into the inner workings of operating systems.
- It offers an invaluable platform for comprehending critical OS components like process management, memory allocation, and file systems.
- The RISC-V iteration of xv6 signifies a notable adaptation, aligning this educational OS with the RISC-V instruction set architecture (ISA).
- XV6, regardless of its traditional or RISC-V variant, deliberately maintains a minimalist approach, eschewing many features found in commercial operating systems.

TOOLS/ API USED

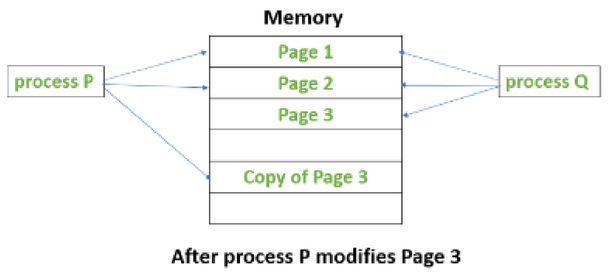
- xv6 Operating System
- System Calls (fork(), sbrk(),brk())
- Memory Management (Page Allocation,Deallocation and Page Tables)
- Concurrency Control (Locks, Semaphores)
- Testing Framework (Tools and Scripts)
- QEMU

COPY ON WRITE

1. Copy-On-Write (COW), a technique devised to alleviate the inefficiencies of memory duplication during process creation.
2. Operates on the principle of deferred copying, postponing the actual duplication of memory pages until a write operation is attempted.
3. Rather than immediately duplicating memory pages upon fork, COW allows both processes to share the same memory pages initially.
4. When either process attempts to modify a shared page, only then is a copy created, ensuring data integrity while minimising unnecessary copying and resource consumption

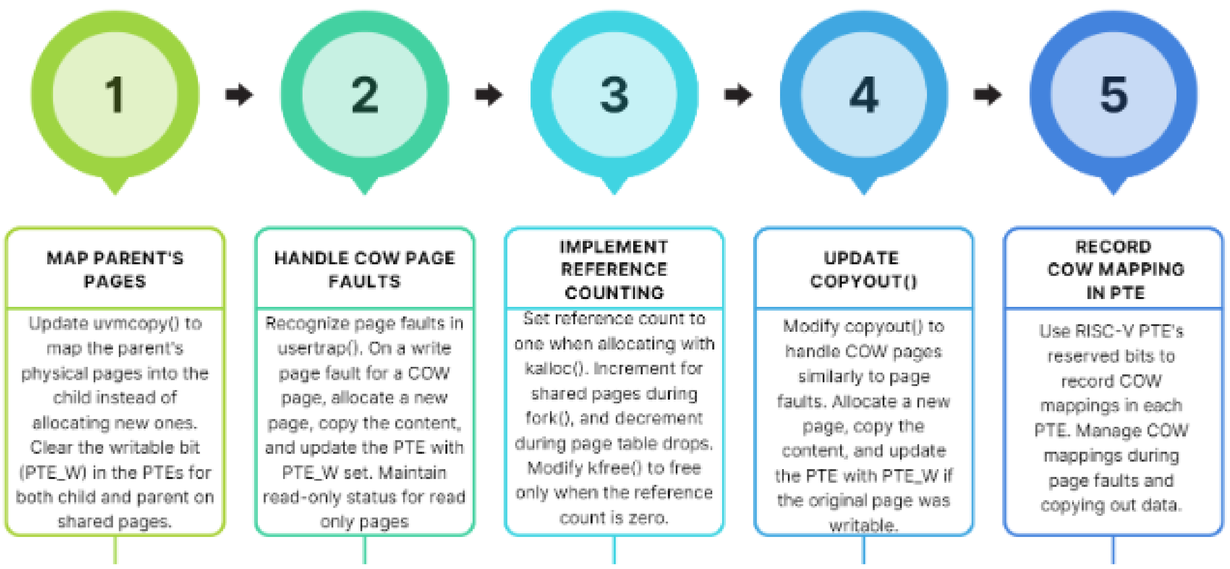


Before process P modifies Page 3



After process P modifies Page 3

METHODOLOGY




```
*pte &= ~PTE_W;


*pte |= PTE_COW;



pa = PTE2PA(*pte);


acquire(&reflock);
reference_counter[pa/PGSIZE]+=1;
release(&reflock);

flags = PTE_FLAGS(*pte);
```

```
{
  else if(r_scause()== 15){
    if ((*my_pte&PTE_COW)){

      uint flags;
      char *mem;


      flags = PTE_FLAGS(*my_pte);
      flags |= PTE_W;
      flags &= ~PTE_COW;

      if((mem = kalloc()) == 0){
        p->killed=1;
        exit(-1);
      }


      memmove(mem, (char*)pa, PGSIZE);


      *my_pte = PA2PTE(mem) | flags;

      kfree((char*)pa);
      if((*my_pte&PTE_COW)&&reference_counter[pa/PGSIZE]==1)
      {

      *my_pte &= ~PTE_COW;
      *my_pte |= PTE_W;
      }
      p->trapframe->epc = r_sepc();
```

APPLICATIONS

Database cloning:
COW allows for creating database clones efficiently. Only data differences are copied, reducing cloning time and resource usage
.

Process Forking:
When a process creates a child process (e.g., using fork in Unix-like systems), both initially share the same memory pages using COW. This saves memory and reduces process creation time, as copying occurs only when necessary (on write).

Snapshots and Versioning:
File systems can leverage COW to create snapshots of directories or files. This allows for reverting to previous versions, recovering from accidental changes, or managing file versions efficiently.

OUTPUT

The successful execution of the cowtest confirms the correct implementation of Copy-On-Write (CoW) memory management in the xv6 operating system.

```
xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ cowtest
ok
 COW TEST PASSED
```

CONCLUSION

In conclusion, the introduction of Copy-on-Write (CoW) presents a demonstrably effective technique for optimising memory management.

This mechanism hinges on the principle of delaying memory duplication until a write attempt is detected. This approach not only minimises memory consumption during the initial fork event, but also extends these benefits throughout the processes' lifecycles.

The implementation of CoW in xv6 serves as a compelling illustration of the benefits derived from leveraging architectural features, such as RISC-V's memory management capabilities, for efficient process creation.

ACKNOWLEDGEMENT

We would like to thank Dr.Jyoti Shetty, our project guide, for giving us the required direction and assistance throughout the project. Ma'am's insightful comments and recommendations were very helpful in forming this endeavor.

We express sincere gratitude to our beloved HOD, Dr. Ramakanth for his appreciation towards this Experiential Learning Project work.

Lastly, we take this opportunity to thank our family members and friends who provided all the backing and support throughout the project work.

TEAM MEMBERS

SUNDARAKRISHNAN N
1RV22CS210

TANMAY UMESH
1RV22CS217

TARUN BHUPATHI
1RV22CS218