Project 04

Implement Copy-on-Write on xv6

Due date:
2024. 06. 09. 11:59pm

xv6 Memory Management

- OS에서 효율적인 메모리 관리는 필수이며, 주로 가상 메모리를 사용해 메모리를 관리합니다.
  paging, segmentation, automatically extending stacks 등의 기술이 있으며,
  이를 통해 운영체제가 주어진 자원을 더 효율적으로 사용할 수 있도록 합니다.

- xv6에서는 기본적인 paging 기술을 이용해 가상 메모리 시스템을 구현하고 있습니다.

- 이번 과제에서는 이론 시간에 배운 Copy-on-Write를 xv6에 구현해보는 것을 목적으로 하며,
  이러한 과정에서 물리 주소 및 페이지 테이블을 어떻게 관리하는지 분석해봅시다.

Default fork()

- fork() 시스템 콜을 사용하면 부모 프로세스를 복제해서 새로운 프로세스를 생성합니다.
- 자식 프로세스와 부모 프로세스가 동일한 context 값(pc, registers. etc)을 가집니다.
  - 메모리 내용은 같아도 서로 다른 메모리 공간에서 실행되므로 한 프로세스가 메모리 일부에 쓰기를 수행해도 다른 쪽 프로세스의 메모리에는 영향을 미치지 않습니다.
- 자세한 내용은 수업자료를 참고하시길 바랍니다.

Copy-on-Write

- 일반적으로 프로세스는 fork() 이후 exec()를 호출해 새로운 프로그램을 실행합니다.
  이렇게 되면 부모로부터 복사해온 페이지들이 쓸모 없게 되어버립니다.

- 메모리의 낭비를 줄이기 위해 Copy-on-Write 기술을 사용할 수 있습니다.

- xv6에서 구현할 사항들은 다음과 같습니다.
  - Copy-on-Write fork
    1. Initial Sharing
    2. Make a copy on write operation

- git clone을 새로 받은 후 과제를 진행하시길 바랍니다.

Specification – CoW (Initial Sharing)

- CoW fork의 올바른 구현을 위해서는, 물리 페이지의 참조 횟수를 알아야 합니다.
  - 페이지의 참조 횟수는 해당 페이지를 가상 주소 공간에 매핑한 프로세스의 수를 의미합니다.
  - 페이지의 참조 횟수는 free page가 어떤 프로세스에 의해 할당될 때 1로 설정됩니다.
  - 추가 프로세스(e.g., 자식 프로세스)가 이미 존재하는 페이지를 가리킬 때, 참조 횟수가 증가해야 합니다.
  - 프로세스가 더 이상 페이지를 가리키지 않을 때(e.g., 페이지 테이블에서 해당 페이지를 가리키지 않을 때), 참조 횟수가 감소해야 합니다.
  - 페이지의 참조 횟수가 0일 때에만 페이지를 free하고 freelist로 반환할 수 있습니다.
  - 페이지들의 참조 횟수를 추적할 수 있는 데이터 구조를 추가해야 합니다. (in kalloc.c)
  - 적절한 locking을 사용해 참조 횟수를 증가 및 감소시키는 함수를 추가해야 합니다.

※ Tip: mmu.h에 정의된 여러 definition이나 macro를 이해하면 좋습니다. (e.g., 가상 주소에서 페이지 번호 추출)

Specification – CoW (Initial Sharing)

- 자식 프로세스를 생성(fork)할 때, 부모의 페이지를 복사해서 자식에게 제공해서는 안됩니다.
- 대신, 자식 프로세스는 새로운 페이지 테이블을 할당 받고, 부모와 자식의 페이지 테이블은 모두 동일한 물리 페이지를 가리켜야 합니다.
- 다수의 페이지 테이블에서 같은 물리 페이지를 매핑하게 되므로, 페이지의 참조 횟수를 증가시키는 함수를 호출해야 할 수 있습니다.
- 이와 같이 부모와 자식이 메모리의 페이지를 공유할 때, 페이지는 읽기 전용으로 표시되어야 합니다. 이는 해당 페이지에 대한 쓰기 접근이 발생할 경우 커널로 trap 되도록 하기 위함입니다.
- 부모의 페이지 테이블이 변경(w.r.t 페이지 권한)되었기 때문에, 페이지 테이블을 재설치하고 CR3 레지스터에 페이지 테이블 포인터를 다시 선언하여 TLB 항목들을 flush 해야합니다.
  - 해당 내용은 **lcr3(V2P(pgdir))** 함수를 호출하여 수행할 수 있습니다.
  - 과제 수행 중 페이지 테이블 항목을 수정할 때는 위 함수를 호출해야 합니다.

Specification – CoW (Make a copy)

- 부모 또는 자식 프로세스가 읽기 전용으로 표시된 페이지에 쓰기를 시도할 때, 페이지 폴트가 발생합니다.
- 기본 xv6의 트랩 handling code에서는 현재 T_PGFLT exception을 처리하지 않습니다.
- 페이지 폴트를 처리하기 위한 트랩 handler를 작성해야 합니다.
  - 해당 함수(CoW_handler)는 user memory의 복사본을 만들어야 합니다.
  - 페이지 폴트가 발생하면 CR2 레지스터가 폴트가 발생한 가상 주소를 저장합니다.
    - xv6의 rcr2() 함수를 통해 획득할 수 있습니다.
  - 가상 주소가 프로세스의 페이지 테이블에 매핑되지 않은 잘못된 범위에 속해 있다면 에러 메시지를 출력하고 프로세스를 종료해야 합니다.
  - 그렇지 않다면, 필요한 페이지의 복사를 진행해야 합니다.

※ Tip: trap.c에서는 mappages와 같은 static 함수를 사용할 수 없으므로 vm.c에 함수 작성하는 것을 추천합니다.

Specification – CoW (Make a copy)

- N개의 프로세스가 하나의 페이지를 공유하고 있다면, 처음 (N-1)개의 프로세스가 trap될 때 각각 별도의 페이지 복사본을 받아야 합니다.
- 이후 마지막 프로세스는 페이지를 가리키는 유일한 프로세스가 되므로, trap 됐을 때 단순히 페이지의 읽기 전용 제한을 제거하고 원래 페이지를 사용하면 됩니다.
- 각 프로세스의 페이지 테이블이 새로운 페이지를 가리키게 될 때, 페이지의 참조 횟수를 적절히 수정할 수 있어야 합니다.
- 페이지 테이블 항목을 변경할 때마다 TLB를 flush해야합니다.

Specification – common

- 다음의 함수들은 주어진 명세대로 구현해야 합니다

- **void incr_refc(uint)**
  - 주어진 페이지의 참조 횟수를 1 증가시킵니다.
  - 적절한 locking을 사용해야 합니다.

- **void decr_refc(uint)**
  - 주어진 페이지의 참조 횟수를 1 감소시킵니다.
  - 적절한 locking을 사용해야 합니다.

- **int get_refc(uint)**
  - 주어진 페이지의 참조 횟수를 반환합니다.
  - 적절한 locking을 사용해야 합니다.

- **void CoW_handler(void)**
  - user memory의 복사본을 만들어야 합니다.

Required system calls

- 다음의 시스템 콜들은 주어진 명세대로 구현해야 합니다.

- **int countfp(void)**
  - 시스템에 존재하는 free page의 총 개수를 반환합니다.

- **int countvp(void)**
  - 현재 프로세스의 user memory에 할당된 가상 페이지(logical page)의 수를 반환합니다.
  - 가상 주소0에서부터 프로세스의 struct proc에 저장된 가상 주소 공간 크기까지의 가상 페이지 수를 세어야 합니다. (커널 주소 공간에 매핑된 페이지는 세지 않아도 됩니다)

- **int countpp(void)**
  - 현제 프로세스의 페이지 테이블을 탐색하고, 유효한 물리 주소가 할당된 page table entry의 수를 반환합니다.
  - xv6에서는 demand paging을 사용하지 않으므로, countvp()의 결과와 동일해야 합니다.

Required system calls

- 다음의 시스템 콜들은 주어진 명세대로 구현해야 합니다.

- **int countptp(void)**
  - 프로세스의 페이지 테이블에 의해 할당된 페이지의 수를 반환합니다.
  - 프로세스 내부 페이지 테이블을 저장하는데 사용된 모든 페이지와 페이지 디렉토리에 사용된 페이지도 포함해야 합니다.
  - user level의 PTE 매핑을 저장하는 페이지 테이블 뿐 아니라 커널 페이지 테이블 매핑을 저장하는 페이지 테이블도 포함되어야 합니다.

Evaluation

- 평가 기준은 다음과 같으며, 각 명세의 일부분을 완성하였다면 부분 점수가 주어집니다.

Evaluation Criteria	Points
Initial Sharing	40
Make a Copy	40
Wiki	20
Total	100

Evaluation

- **Completeness** : 명세의 요구조건에 맞게 xv6가 올바르게 동작해야 합니다.
- **Defensiveness** : 발생할 수 있는 예외 상황에 대처할 수 있어야 합니다.
- **Wiki & Comment** : 테스트 프로그램과 위키를 기준으로 채점이 진행되므로, 위키는 최대한 상세히 작성되어야 합니다.
- **Deadline** : 마감 기한을 반드시 지켜야 하며, 마감 기한 이전에 마지막으로 제출된 코드를 기준으로 채점됩니다.
- **DO NOT SHARE AND COPY !!**

Wiki

- Wiki는 아래 요소들을 포함해야 합니다
- **Design** : 명세에서 요구하는 조건에 대해서 **어떻게 구현할 계획**인지, 어떤 자료구조와 알고리즘이 필요한지, **자신만의 디자인**을 서술합니다.
- **Implement** : 실제 구현 과정에서 **변경하게 되는 코드영역**이나 **작성한 자료구조** 등에 대한 설명을 구체적으로 서술합니다.
- **Result** : 컴파일 및 **실행 과정**과, 해당 명세에서 요구한 부분이 정상적으로 동작하는 **실행 결과**를 첨부하고, 이에 대한 동작 과정에 대해 설명합니다.
- **Trouble shooting** : 과제 수행 도중 **발생했던 문제**와 이에 대한 **해결 과정**을 서술합니다. 문제를 해결하지 못했다면 **어떻게 해결하려 하였는지**에 대해 서술합니다.
- **and whatever you want :)**

Submission

- 구현한 코드와 wiki를 LMS 과제란에 제출합니다.
- WiKi 파일과 xv6-public 파일 전체를 한 디렉토리에 포함시킨 후, 압축하여 제출합니다.
- 제출할 디렉토리의 이름은 "OS_project04_수업번호_학번" 입니다.
- Wiki 파일의 이름은 "OS_project04_수업번호_학번_이름.pdf" 입니다.
  - 제출할 디렉토리의 양식 예시는 다음과 같습니다.
    - ex) OS_project04_12345_2022123123

      -- xv6-public

      -- OS_project04_12345_2022123123_NAME.pdf
    - 해당 디렉토리를 압축하여 OS_project04_12345_2022123123.zip 파일을 제출합니다.
- **제출 기한: 2024년 6월 9일 11:59 pm**
- **추가 제출 기한: 2024년 6월 10일 11:59 pm (Penalty of 50% of the total score)**

Thank you