10장 가상메모리 과제

1173101 김민경

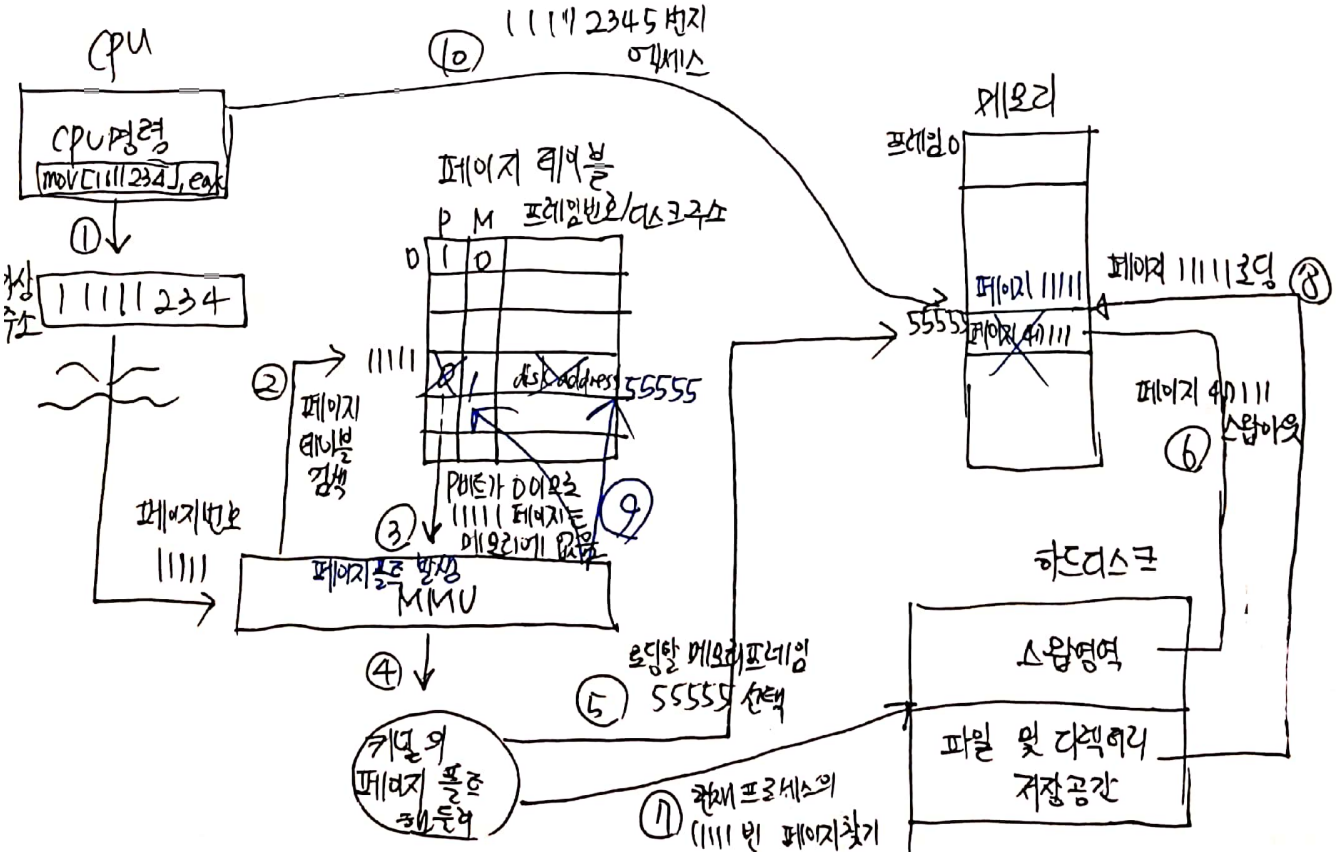
1. 요구 페이징 기법을 정의하고 목적은 무엇인지 함께 기술하라.

요구 페이징은 물리메모리의 크기 한계를 극복하기위해 페이징 기법을 기반으로 하여 만들어진 가상 메모리 기법이다. 이는 프로세스 실행 시 시작하는 자동할당되는 첫 페이지를 제외하고 나머지 페이지는 필요할 때 물리메모리를 할당 받아 디스크로부터 로딩한다. 요구 페이징은 물리메모리의 크기 한계를 극복하여 물리메모리보다 큰 프로그램이나 여러 프로그램을 동시에 실행시키는 가상 메모리 기법을 구현하는것이 목표이다.

2. dirty page란 어떤 페이지를 뜻하는가? 어떻게 구분하는가?

dirty page는 메모리에 적재된 페이지 중 수정이 가해진 페이지를 뜻한다. 이는 modified page 라고도 불린다. 페이지 테이블의 modified/dirty bit 값으로 구분하여 1일 경우 dirty page이다.

3. 페이지 폴트는 언제 발생하는가? 발생할 때 페이지 폴트 핸들러(커널)가 새로운 페이지를 가지고 오는 전체과정을 그림으로 그리고 자세히 설명하라.



주 메모리에 할당하고자 하는 페이지를 교체하는 과정에서 CPU가 참조하고자 하는 페이지가 물리 메모리에 복제하는 경우에 페이지 폴트가 발생한다

① TLB 미스가 발생하여 MMU에서 페이지 테이블을 검색한다.

② 해당 항목의 presence bit가 0 임을 확인 한 후, 하드웨어 신호인 페이지 폴트 예외가 발생되어 커널의 페이지 폴트 핸들러가 실행된다.

③ 운영체제가 임의의 프레임을 선정하여, 해당 프레임에 실행 파일 혹은 스왑 영역에서 읽으려는 페이지를 찾아 스왑-인 한다. 이때, 만일 해당 프레임에 다른 프레임이 존재하고 변경사항이 존재한다면, 스왑-아웃 하여 변경 사항을 저장해준다.

④ 페이지 테이블에 프레임 번호를 기록하고 presence 비트를 1로 수정한다.

⑤ 페이지 폴트 핸들러가 종료되고, 기존 명령이 재개된다.

4. 쓰기 시 복사는 어떤 목적으로 제안된 기법인가? 그 과정을 빌로 간단히 설명하라.

프로세스들 중 같은 자원을 공유하는 경우, 즉 자식 프로세스가 메모리를 복사하지 않고 부모의 메모리를 공유하며 프로세스 별 자원을 수집하게 되는 경우에, 메모리 쓰기가 발생한 페이지만 새로 할당해주는 쓰기 시 복사 방법은 자원과 동일한 자식 프로세스를 만드는 기존의 완전 복사 기법에 비하여 메모리와 프로세스 생성 시간을 절약할 수 있어 한정된 자원공간에서 효율적인 자원을 공유할수 있다는 장점이 있다.

5. 운영체제는 커널이 사용하는 메모리에 대해서는 요구 페이징을 사용하지 않는다. 그 이유는 무엇인가?

커널 코드와 데이터는 항상 물리메모리에 있어야 하므로 요구 페이징을 사용하지 않는다. 예를 들어 인터럽트 핸들러가 담긴 페이지가 물리 메모리에 있지 않고 스왑-아웃되어 하드 디스크에 존재한다면, 인터럽트가 발생하였을 때 인터럽트 핸들러가 실행될 수 없거나 실행되는데 매우 긴 시간이 걸린다.

6. 빈 프레임이 없는경우, 프레임에서 교체할 페이지 프레임을 선택해야한다. 이때 교체 범위에 따라 나뉘는 2가지 방법을 말하고, 2 중 어떤것이 더 많이 활용되는가?

지역교체는 페이지 로딩을 요청한 프로세스에게 할당된 프레임 중 희생 프레임을 선택하는 방법이다. 다른 프로세스에게 할당된 프레임을 건드리지 않아 스레싱이 다른 프로세스로 전파되지 않는 장점이 있다.

전역교체는 프로세스에 상관없이 전체 메모리 프레임 중에서 희생 프레임을 선택하는 방법이다. 이 가운데 전역교체는 지역 교체보다 페이지 폴트을 덜 유발시켜서 대부분 전역 교체를 사용한다.

7. 페이지 테이블 엔트리에는 물리 프레임 번호 외에 valid bit, modified bit, use bit (reference bit) 등으로 구성된다. 각 비트가 0인 경우와 1인 경우에 따라 그 의미를 간단히 설명하라.

valid bit는 1일 경우 페이지가 메모리 프레임에 존재한다. 0일경우는 메모리 프레임에 존재하지 않는다.

modified bit는 1일 경우 페이지가 로딩된 후 수정되었을 경우를 말한다. 0일경우는 변경사항이 없는 경우를 말한다.

reference bit는 1이면 페이지가 최근에 참조된 것을 나타낸다. 0이면 역시 페이지가 참조되지 않은것이다.

8. 스레싱을 정의하고, 스레싱이 발생하는 원인을 설명하고, 스레싱의 해결책을 제시하라.

스레싱은 페이지 폴트가 계속적으로 발생하여 메모리 프레임에 페이지가 반복적으로 교체되는 현상을 뜻한다. 이 때 디스크 입출력이 심각하게 증가해 프로세스의 응답시간이 길어지고 CPU 활용률은 감소한다. 스레싱이 발생하는 원인은 메모리 양에 비해 실행중인 프로세스의 개수가 과한 경우, 특정 시간에 너무 많은 프로세스가 실행되는 경우 등으로 다양하다.

해결책으로는 몇몇 프로세스를 강제로 종료시켜 다중 프로그래밍의 정도를 낮추는 방법이 있다. 그리고 작업 집합을 사용하여서도 해결이 가능하다. 스레싱의 발생 원인은 고정된 페이지 프레임이 충분하지 않기 때문에 발생하는 것인데, 지역성을 기반으로 참조가 빈번한 프레임들을 작업 집합으로 고정시켜 페이지 폴트를 줄여 스레싱 현상을 예방할 수 있다. 이 외에도, 다중 프로그래밍 정도의 시스템 허용 최대치를 낮추어 설정하거나 물리 메모리량을 높이는 방법이 존재한다.

9. 스레싱이 발생하고 있는지 어떤 지표 (CPU 활용률)의 변화로 알 수 있는가?

동시에 실행되는 프로세스의 개수가 많을수록 CPU 유휴시간이 줄어 활용률이 증가되어야 하는데, 만약 정도 지나쳐서 각 프로세스에게 할당되는 메모리 프레임 개수가 작아져서 페이지 폴트가 빈번하게 일어나게 되는 스레싱은 디스크 입출력으로 인해 CPU 활용률이 오히려 떨어진다. 따라서 동시에 실행되고 있는 프로세스의 개수가 많음에도 불구하고 CPU 활용률이 오히려 떨어질 때를 스레싱이 발생한 상태로 판단할 수 있다.

10. 프로그램의 실행 초기에 페이지 폴트가 계속 발생할 것이다.

이 페이지 폴트의 발생과 작업집합과의 관계는 무엇인가?

작업 집합은 일정 시간 범위 내에 프로세스가 액세스한 페이지들의 집합이며, 프로세스가 실행 중 계속되는 페이지 폴트는 프로세스의 작업 집합을 형성하고 있다고 볼 수 있다. 실행 초기에는 CPU가 참조하고자 하는 페이지의 집합을 알 수 없어서 페이지 폴트가 발생하지만 실행 후 시간이 지나면서는 지역성을 기반으로 참조되는 페이지들이 작업 집합 형태로 고정되어 페이지 폴트 발생이 줄어든다.

11. 스레싱, 페이지 폴트, 작업집합이 3단어를 쓰이 하나의 스토리를 문장을 만들어보라.

작업집합에 포함된 페이지들이 충분히 메모리 프레임에 적재되지 않아 페이지 폴트가 많이 일어나면 디스크 입출력이 많이 일어나 CPU 활용률이 줄어드는 스레싱이 일어난다.

12. 참조의 지역성을 정의하고, 참조의 지역성을 활용하는 사례를 하나 들어라.

참조의 지역성은 CPU가 프로그램을 실행하는 동안 짧은 시간 범위 내에 일정구간의 메모리 영역을 반복적으로 엑세스하는 경향을 말한다.
참조의 지역성을 이용함으로서 운영체제는 현재 실행중인 프로세스가 가까운 미래에 어떤 페이지를 엑세스할 것인지 합리적으로 예측할수 있고 이에 따라 현재 메모리 프레임이 존재하는 것 중 가까운 미래에 사용될 페이지가 스왑-아웃 되지 않도록 정책을 펴서 미래의 페이지 폴트를 줄일수 있다.

13. 프로세스가 페이지를 로드하기 위해 메모리 프레임을 요청할 때, 메모리 프레임을 할당하는 전략은 시스템의 성능을 결정하는 중요한 변수이다. 이유가 무엇인가?

프레임 할당의 목표는 프로세스의 작업 집합에 포함된 페이지가 충분히 메모리에 할당되도록하여 페이지 폴트를 줄이고 스레싱이 발생하지 않도록 하는데 있다. 어떠한 프로세스의 크기와 관계없이 모든 프로세스에게 동일한 개수의 메모리 프레임을 할당하는 균등할당과 프로세스의 크기에 비례하여 메모리 프레임의 개수를 할당하는 비례할당 방식이 있다. 균등할당은 알고리즘이 단순하지만 작은 프로세스가 필요 이상의 많은 프레임을 할당받거나 큰 프로세스가 프레임의 개수가 부족하여 페이지 폴트가 자주 발생하는 단점이 있다. 하지만 비례할당 방식은 이와 달리 페이지 폴트가 적으나 프로세스의 크기를 실행 전이나 실행중 명확히 알기 어렵다는 현실적 문제가있다.

14. 작업 집합과 운영체제의 프레임 할당 정책과는 어떤 관계가 있는지, 스레싱과 관련하여 설명해보라.

운영체제의 프레임 할당 정책은 스레싱이 일어나지 않도록 한 프로세스에 할당되는 메모리 프레임의 개수를 적절히 조절해야한다. 프레임 할당이 부족한 경우 스레싱이 일어나고 반대로 프레임 할당이 과도할 경우 메모리가 낭비된다. 따라서 프로세스에게는 작업 집합에 포함된 페이지를 로딩할 수 있을 정도의 프레임을 할당하는 것이 적당한데, 작업 집합은 프로세스의 실행과정에서 시간에 따라 가변적이므로 한 프로세스에게 할당해 줄 프레임의 적정 개수는 작업 집합을 약간 넘나드는 수준으로 설정해주어야 적합하다.

15. 페이지 교체 알고리즘의 목표는 무엇인가? 최적, FIFO, LRU, 페이지 교체 알고리즘을 간단히 설명하고, 그 중 가장 성능이 좋은 것과 현재 가장 많이 사용되는 것은 무엇인가?

최적 교체 알고리즘은 미래에 사용될 가능성이 가장 낮은 페이지를 희생 페이지로 선택한다. FIFO에 비해 페이지 폴트가 적어지고 최고의 성능을 낸다는 장점이 있으나 페이지 호출 순서를 사전에 파악해야하므로 앞으로 발생할 사례를 예측하는 것을 할 수 없기에 비현실적이다.

FIFO 교체 알고리즘은 FIFO 큐에서 가장 먼저 들어온 페이지를 가장 먼저 교체하고 페이지 교체를 위해 time stamp 를 사용한다. 간단한 알고리즘을 사용한다는 장점이 있지만 오랫동안 사용된 페이지가 교체될 확률이 높기 때문에 페이지 폴트 발생률이 증가된다는 단점이 있다.

LRU 알고리즘은 각 페이지별 time stamp 카운터를 지정하여 교체시점에서 가장 오랫동안 사용되지 않은 페이지와 교체하는 기법이다. 이는 시점 별 가장 오랫동안 사용되지 않은 페이지와 교체하여 유동적인 페이지 교체가 가능하다는 장점이 있지만 구현이 복잡하고 하드웨어 지원과 커널 모드의 주기적인 실행 등 오버헤드가 따른다.

Clock LRU- LRU와 FIFO를 섞은 알고리즘으로, 참조 비트와 하드웨어를 사용하여 원형 큐에서 포인터로 참조 비트를 변환하는 구성을 띤다. LRU에 비하여 모든 프레임의 참조 비트를 0으로 만드는 과정이 없기 때문에 오버헤드가 상대적으로 작다.

이 중 할당된 프레임 수가 고정된 경우의 최적 교체 알고리즘의 페이지 부재가 가장 적기 때문에 성능이 가장 좋다. 하지만 최적 교체 알고리즘은 페이지 호출 순서를 사전에 파악하기 위한 기능 구현이 비현실적이기 때문에 다른 알고리즘의 성능을 평가하는 기준으로서 의의를 가지며, 실현 가능하며 제일 성능이 좋아 가장 많이 사용되는 알고리즘은 Clock LRU 알고리즘이다.