- **10. 페이지 교체 알고리즘[149]**
  - **개관**
    - **페이징 기법**
    - **학습목표**
      - **1. 다양한 페이지 교체 알고리즘을 이해**
      - **2. 프로세스별 페이지 집합 관리를 이해**
    - **주요용어**
      - **FIFO**
      - **Belady의 이상현상**
      - **LRU**
      - **국부성**
      - **LFU**
      - **2차 기회**
      - **워킹 세트**
      - **쓰레싱(thrashing)**
      - **PFF**
  - **10.1. 페이지 교체 알고리즘[149]**
    - **페이징 기법에서는 모든 페이지 프레임이 사용되고 있는 것이 일반적**
      - **운영체제의 역할**
        - **메모리에 새로 적재되어야 할 페이지를 위해 적절한 교체 대상 페이지 프레임을 선택하여 그 내용을 보조기억장치에 보관한 후 새로운 페이지를 적재해야 한다**
    - **최적화 원칙 (principle of optimality)**
      - **최적의 성과를 얻기 위해 페이지 프레임에 있는 페이지 중 이후로 가장 오랫동안 사용되지 않을 페이지를 교체 대상으로 선택**
      - **이론적, 미래 예측X로 실제 구현 불가**
      - **다른 페이지 교체기법이 얼마나 최적성을 갖고 있는지 비교**
    - **효율적인 동작을 위해 교체가 일어나지 않게 할 필요가 있는 페이지(불변 페이지)**
      - **커널 코드 영역**
      - **보조기억장치 드라이버 영역**
      - **시간을 맞춰 동작해야 하는 코드 영역**
      - **데이터 버퍼 영역: DMA(Direct Memory Access) 등에 의해 입출력장치로부터 직접 데이터가 교환되어야 하는 영역**
  - **10.1. 페이지 교체 알고리즘(2) [150]**
    - **1. FIFO 페이지 교체**
      - **FIFO(First-In First-Out) 페이지 교체 알고리즘**
        - **메모리 내에 가장 오래 있었던 페이지를 교체 대상으로 선택하여 교체**
        - **메모리에 존재하지 않는 페이지가 도착하면 큐의 끝부분(tail)에 두고, 큐의 앞부분(head)에서 교체할 페이지를 정함**
      - **단점**
        - **실제 메모리에 가장 오래 있었던 페이지는 앞으로도 계속 사용될 가능성이 높기 때문에 가장 많이 쓰는 페이지를 교체 시킬 가능성이 있음**
        - **Belady의 이상현상(anomaly)**
          - **프로세스에 더 많은 수의 페이지를 할당할 경우 오히려 페이지 부재가 더 많이 발생할 수 있다**
          - **실제로 하나의 프로세스에 4개의 페이지 프레임을 할당했을 경우, 3개의 페이지 프레임을 할당했을 때보다 페이지 부재가 한 번 더 발생했음을 알 수 있다.**
          - **원인**
            - **(1) 오래된 페이지가 중요한 페이지일 수도 있음**
            - **(2) 페이지 참조 패턴과 프레임 개수의 비효율적인 관계**
            - **(3) "오래된 페이지가 불필요한 페이지"라는 보장이 없음**
    - **2. LRU 페이지 교체**
      - **LRU(Least Recently Used) 페이지 교체 알고리즘**
        - **메모리 내에서 가장 오랫동안 사용되지 않은 페이지를 교체 대상으로 선택하여 교체**
        - **국부성 휴리스틱**
          - **최근의 상황이 가까운 미래에 대한 좋은 척도이다.**

- 국부성(localty)
  - 프로세스가 기억장치 내의 정보를 균일하게 액세스하는 것이 아니라 어느 한순간에 특정 부분을 집중적으로 참조하는 현상
  - 시간 국부성
    - 처음에 참조된 기억장소가 가까운 미래에도 계속 참조될 가능성이 높다
    - ex) 반복문이 특정 서브프로그램 2개를 호출 (페이지는 A, B) => 연속된 페이지가 차례로 참조
  - 공간 국부성
    - 일단 하나의 기억장소가 참조되면 그 근처의 기억장소가 계속 참조되는 경향이 있다
    - ex) 대용량 데이터 변수 저장 후 순차적 처리 => 연속된 페이지가 차례로 참조
- 방법
  - 참조 시간 이용
    - 각 페이지가 참조될 때마다 그 때의 시간을 테이블에 기록
    - 교체가 필요한 경우 참조시간이 가장 오래된 페이지가 교체 대상으로 선택
  - 리스트 이용
    - 메모리에 적재된 페이지 번호를 저장하는 리스트를 이용하여 페이지를 액세스하면 해당 페이지 번호를 리스트의 선두에 옮긴다.
    - 교체가 필요한 경우 리스트에 끝에 있는 페이지가 교체 대상으로 선택된다.
- 특징
  - 장점
    - (1) Belady의 이상현상이 발생하지 않는다.
    - (2) 많은 경우 최적화 원칙에 부합한다.
  - 단점
    - (1) 하나의 프로세스가 여러 페이지로 구성되는 커다란 루프를 가지고 있을 때 가장 오랫동안 사용되지 않는 페이지가 가장 가까운 장래에 사용될 페이지가 되어 즉시 페이지 부재 발생
    - (2) LRU는 막대한 오버헤드를 초래
      - 매번 시간을 기록하고 가장 오래된 참조시간을 찾는 것
      - 리스트를 유지하는 것

10.1. 페이지 교체 알고리즘 (3) [154]

3. LFU 페이지 교체

LFU(Least Frequently Used) 페이지 교체 알고리즘

- 메모리 내에서 참조된 횟수가 가장 적은 페이지를 교체 대상으로 선택하여 교체

특징

- 단점
  - 가장 드물게 이용되는 페이지가 가장 최근에 메모리로 옮겨진 페이지일 가능성
  - 초기에 매우 많이 사용된 후 더 이상 사용되지 않는 페이지는 교체 대상에서 제외됨으로써 불필요하게 메모리를 점유할 가능성이 있다.
  - 오버헤드가 크다.
    - (1) 매번 참조 횟수를 증가시키고 (2) 가장 적은 참조 횟수를 찾는 경우

4. 2차 기회(second chance) 페이지 교체

1) 참조 비트가 0이면서 2) 메모리 내에 가장 오래 있었던 페이지를 교체 대상으로 선택하여 교체

FIFO 보완

참조 비트

- 페이지 프레임에 있는 페이지마다 부여된 값으로, 해당 페이지가 메모리에 적재될 때는 0이지만 계속 메모리에 있는 상태에서 추가로 해당 페이지가 참조되면 1이 된다.
- 참조 비트를 0으로 만드는 상황 존재

방법

- 1. 큐의 선두 항목을 꺼내 참조 비트를 조사
- 2. 참조 비트가 1이면, 0으로 바꿔 큐의 뒤에 추가하고 1단계로 돌아간다.
- 3. 만약 참조 비트가 0이면, 그 페이지를 교체 대상으로 선택
- 참조할 페이지가 메모리에 존재하면?
  - 큐의 위치는 변화시키지 않고 해당 페이지의 참조 비트만 1로 설정

클럭(clock) 페이지 교체 알고리즘

- 원형 큐가 시곗바늘이 돌아가는 것처럼 관리
- 2차 기회 페이지 교체 알고리즘은 큐가 꽉 찬 경우에만 삭제가 발생하기 때문에 (head, tail)이 아닌 하나의 포인터만으로 구현 가능
  - 포인터는 항상 마지막에 추가된 페이지의 다음 위치를 가리킴
- 방법
  - 클럭 페이지 교체에서 참조할 페이지가 존재하는 경우
    - 포인터 위치 변화 X, 참조비트만 1로 설정
  - 클럭 페이지 교체에서 참조할 페이지가 존재하지 않는 경우

10.2. 프로세스별 페이지 집합관리 [159]

- 프로세스별 페이지 집합
  - 프로세스마다 사용할 수 있는 페이지 프레임의 개수만큼 페이지들을 메모리에 유지할 수 있음
  - 집합의 크기가 작을수록?
    - 메모리에 적재할 수 있는 프로세스의 수는 많아져서 시스템 처리량이 증대될 수 있음
    - 각 프로세스별 페이지 부재는 자주 발생할 수 잇어 성능이 저하될 수 있음
  - 집합의 크기가 클수록?
    - 메모리에 적재될 수 있는 프로세스의 수는 줄어들지만 프로세스별 페이지 부재는 덜 발생한다
- 1. 워킹 세트 알고리즘
  - 워킹 세트(working set)
    - 한 프로세스가 최근에 참조한 페이지의 집합
      - 국부성(Locality)를 활용하여, 프로세스가 특정 기간 동안 자주 사용하는 페이지들을 유지하면서 페이지 폴트를 줄이는 것 목표
    - W(t, τ): 시간 t에서의 워킹 세트
      - τ: 관찰할 시간 윈도우 크기
      - p_i: τ 동안 참조된 페이지
      - 시각 t - τ + 1부터 시각 t까지 참조한 페이지의 집합
    - 예시
      - t = 4, τ = 3
    - 특징
      - 프로세스의 워킹 세트는 프로세스가 수행됨에 따라 페이지가 삭제되기도 하고 추가되기도 하며, 변함없이 유지되기도 한다.
      - 윈도우 크기가 고정되어 있더라도 워킹 세트의 크기는 달라질 수 있다.
      - 쓰레싱(thrashing)
        - 페이지 부재가 비정상적으로 많이 발생하여 프로세스 처리보다 페이지 교체처리에 너무 많은 시간을 소비함으로써 시스템의 처리량이 급격히 저하되는 현상
    - 워킹 세트 알고리즘
      - 프로세스의 워킹 세트를 메모리에 유지시키는 것을 원칙으로 하는 기법
      - 프로세스마다 워킹 세트의 크기에 맞게 페이지 프레임의 개수를 조절
      - 운영체제의 역할
        - 운영체제는 각 프로세스의 워킹 세트를 감시
        - 충분한 여분의 페이지 프레임이 존재하면 다른 프로세스를 들여와 실행 프로세스의 수를 늘린다.
        - 실행 중인 프로세스들의 워킹 세트 크기의 합이 증가하여 총 페이지 프레임 수를 넘어서면 운영체제는 우선순위가 가장 낮은 프로세스를 일시적으로 중지시켜 여유 페이지 프레임을 확보한다.
      - 문제점
        - 과거를 통해 미래를 예측하는 것이 정확하지 않음
        - 워킹 세트를 정확히 알아내고 이를 계속적으로 업데이트하는 것이 현실적으로 어려움
        - 워킹 세트를 구하기 위한 윈도 크기 τ의 최적값을 알기 어려우며 이 역시 변화할 수 있음
- 2. PFF 알고리즘
  - PFF(Page Fault Frequecy) 알고리즘
    - 페이지 부재 빈도(PFF)를 이용하여 프로세스별 페이지 집합의 크기를 변화시키는 기법
    - 페이지 부재 빈도(PFF)
      - 페이지 부재로 교체가 발생하는 빈도를 나타내는 척도
      - 페이지 부재가 발생하면 직전 페이지 부재 이후로 경과된 시간의 역수로 정함

- 예) 3초 전 페이지 부재가 발생했으면 PFF는 1/3
- 페이지 부재 빈도의 상한과 하한을 정해두고,
  - 페이지 부재 빈도가 상한보다 높으면
    - 페이지 프레임의 개수를 1 증가
  - 페이지 분재 빈도가 하한보다 낮으면
    - 그 사이에 참조되지 않았던 페이지를 모두 제거하고 페이지 프레임 개수도 줄임
- 장점
  - 프로세스별 페이지 집합이 워킹 세트 알고리즘처럼 자주 발생하지 않음
    - 페이지 부재가 발생하고, 그 때의 페이지 부재 빈도가 상한이나 하한을 벗어나는 경우에만 바뀐다.

- 요약
  1. 페이지 교체는 메모리가 완전히 사용되고 있을 때, 새로 적재되어야 할 페이지를 위해 어느 페이지가 교체되어야 하는지를 다룸
  2. 페이지 교체 알고리즘에는 FIFO, LRU, LFU, 2차 기회 페이지 교체 등이 있음
  3. 최적의 페이지 교체방법은 앞으로 가장 오랫동안 사용되지 않을 페이지를 선택하는 방법이지만 현실적으로 불가능함
  4. 프로세스는 기억장치 내의 정보를 균일하게 액세스하는 것이 아니라 어느 한 순간에도 특정 부분을 집중적으로 참조하는 국부성을 보임
  5. 워킹 세트는 한 프로세스가 최근에 참조한 페이지의 집합
  6. 프로세스가 효율적으로 수행되기 위해서는 워킹 세트가 메모리 내에 유지되어야 함.
  7. PFF 알고리즘의 기본 아이디어는 페이지 부재 빈도가 높으면 페이지 프레임을 해당 프로세스에 더 배정하고 낮으면 회수하는 것