운 영 체 제

4장 가상 메모리 관리
(페이지 교체 알고리즘)

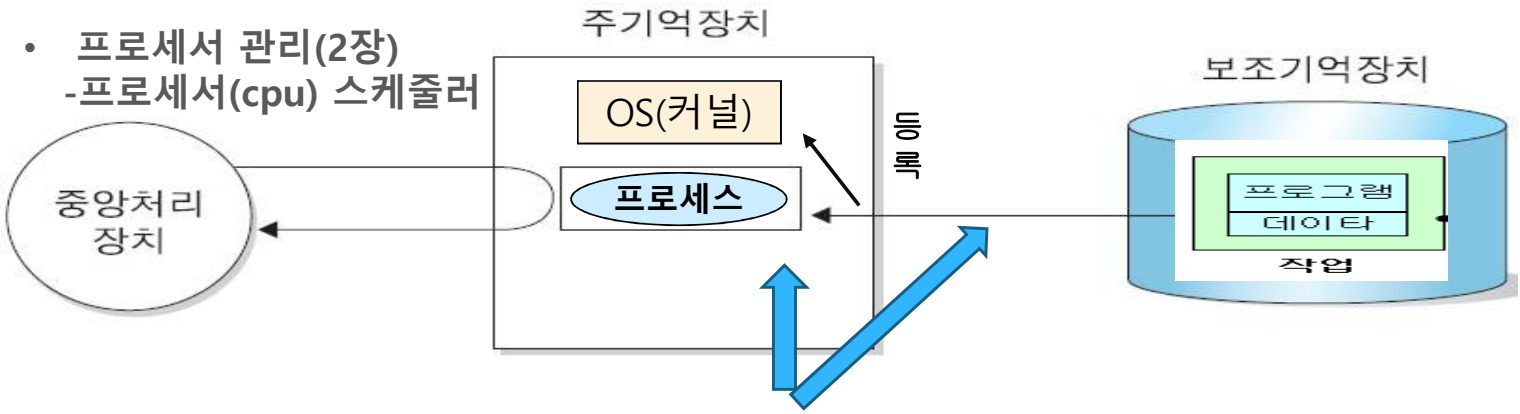
4장 가상 메모리 관리

4.1 개요
4.2 페이징(paging)
4.3 세그먼테이션(segmentation)
4.4 세그먼트/페이징 혼용 기법
4.5 페이지 교체 알고리즘
4.6 스레싱(thrashong)
* 가상 기억장치 기타 관리 사항

학습 내용

- 가상기억장치 관리 기법
- 비용 모델
- 페이지 부재
- 4.5 페이지 교체 알고리즘(1)
  - FIFO(First In First Out) 알고리즘
  - OPT (Optimal Replacement: 최적 교체) 알고리즘
  - LRU(Least Recently Used) 알고리즘
  - LFU(Least Frequently Used) 알고리즘

4장 가상 메모리의 개요



- 프로세서 관리(2장)
  -프로세서(cpu) 스케줄러
- **주기억장치의 관리(3장)**
  -Job 스케줄링
  -주소 바인딩
  -주기억장치 관리 기법
    -인출 기법(요구 인출 기법,예상인출 기법)
    -배치 기법(최초 적합(first-fit), 최적 적합(best-fit) 및 최악 적합(worst-fit))
    -교체 기법
  -주기억장치 할당 기법
    -연속 할당 기법(단일사용자 연속, 고정 분할, 가변 분할)
- 가상 메모리 관리(4장)
  -분산 할당 기법(페이징 기법, 세그먼테이션 기법)
  -페이지 교체 기법

가상기억장치 개념

- 가상기억장치(**virtual storage**)
  - 개념
    - 사용자 프로그램을 여러 블럭들로 분할
    - 가상 기억장치는 보조 기억장치의 일부를 주기억장치처럼 사용하는 것으로 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법
    - 실행시 필요한 블럭들만 비연속적으로 주기억장치에 적재시킴
    - 가상 기억 장치에서 프로그램이 갖는 연속적인 주소가 실제 기억 장치에서는 연속적일 필요가 없다.
    - 가상 기억장치에 저장된 프로그램을 실행하려면 가상 기억장치의 주소를 주기억장치의 주소로 바꾸는 주소 변환 작업이 필요하다.
    - 블록 단위로 나누어 사용하므로 연속 할당 방식에서 발생 할 수 있는 단편화를 해결할 수 있다.
    - 주기억장치의 이용률과 다중프로그래밍의 효율을 높일 수 있다.
  - 블록의 종류에 따라
    - 페이징 시스템(paging system)
      - 사용자 프로그램을 같은 크기의 블럭들로 분할하는 경우
    - 세그먼테이션 시스템(segmentation system)
      - 사용자 프로그램을 서로 다른 크기의 논리적인 단위로 분할하는 경우
    - 페이징 시스템과 세그먼테이션 시스템의 합성 기법

가상기억장치 관리 개념

- 가상기억장치 관리 기법
  - 가상기억장치 시스템의 효율성 및 성능의 향상을 위해 필요한
    - 할당 기법(allocation strategies)
    - 호출 기법(fetch strategies)
    - 배치 기법(placement strategies)
    - 교체 기법(replacement strategies)

가상기억장치 관리 개념

- 가상기억장치 관리 기법
  - 할당 기법(allocation strategies)
    - 각 프로세스들에게 할당해 줄 주기억장치의 양을 결정하는 기법
      - 고정 할당(fixed allocation) 기법,가변 할당(variable allocation) 기법
  - 호출 기법(fetch strategies)
    - 특정 페이지를 언제 주기억장치에 적재할 것인가를 결정하는 기법
      - 요구 호출(demand fetch, demand paging) 기법
      - 예측 호출(anticipatory fetch, prepaging) 기법

가상기억장치 관리 개념

- 가상기억장치 관리 기법
  - 배치 기법(placement strategies)
    - 주기억장치의 어디(빈 공간)에 위치시킬 것인가를 결정하는 기법
    - 페이징 시스템에서의 배치 기법은 불필요함
      - 같은 크기의 페이지 프레임으로 운영하기에 빈 프레임에 어떤 페이지든 적재할 수 있어 배치 기법이 필요 없음
    - 세그먼테이션 시스템에서의 배치 기법 필요
      - 프로세스에 따라 세그먼트 크기가 다르고 빈 공간도 크기가 다르기에 배치 기법이 필요
      - 최초 적합(first-fit) 전략
      - 최적 적합(best-fit) 전략
      - 최악 적합(worst-fit) 전략

가상기억장치 관리 개념

- 가상기억장치 관리 기법
  - 페이지(고정 분할) 교체 기법(replacement strategies)
    - 새로운 페이지가 적재되어야 하는 상황에서 빈 페이지 프레임이 없을 때 어느 페이지를 교체시킬 것인지 결정
      - FIFO(First In First Out) 알고리즘
      - OPT (Optimal Replacement: 최적 교체) 알고리즘
      - LRU(Least Recently Used) 알고리즘
      - LFU(Least Frequently Used) 알고리즘

비용 모델

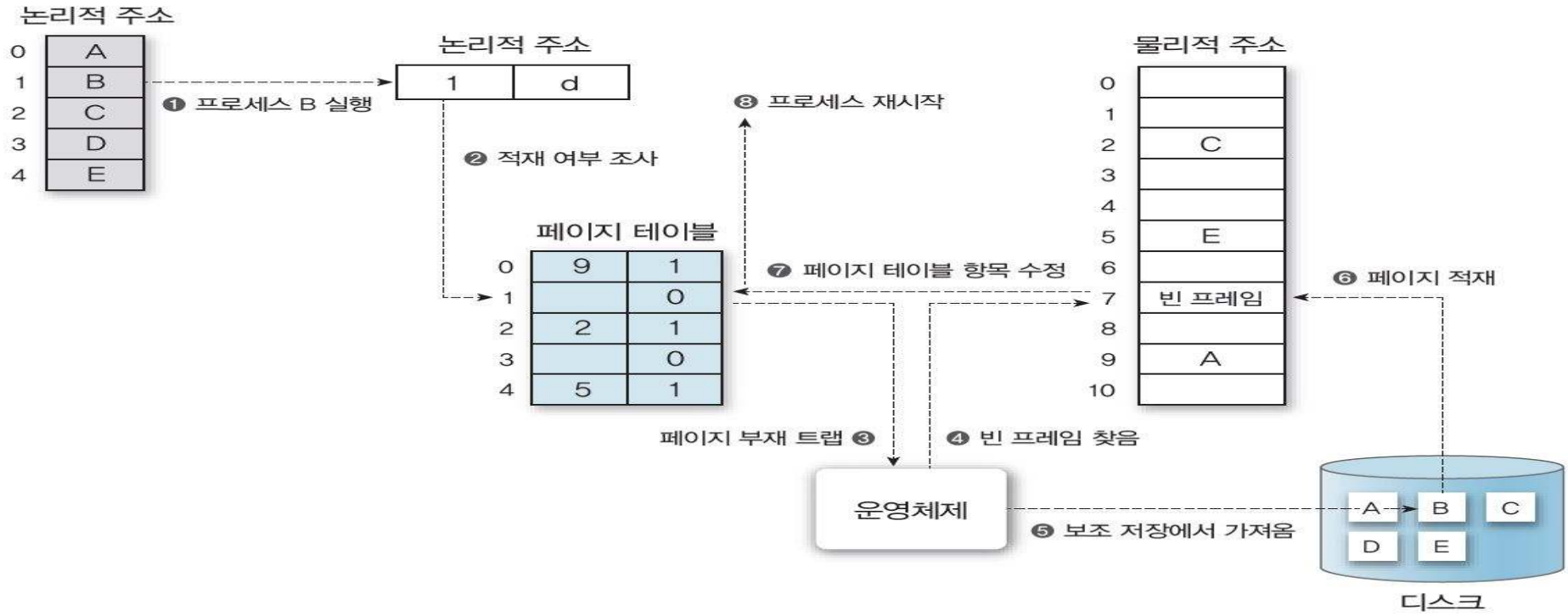
- 가상기억장치 시스템의 비용 모델
  - 페이지 부재(page fault) 발생 빈도
  - 페이지 부재 발생율(page fault rate)

  - 운영비용, 즉, 각 프로세스들이 실행 중에 발생시키는 페이지 부재율을 최소한으로 줄일 수 있도록 설계되어야 함
  - Context switching이나 커널의 개입을 최소화
  - 시스템 성능의 향상

페이지 부재

페이지 부재의 개념

- 프로세스가 존재 비트로 표시된 페이지에 액세스하지 않는다면 존재 비트 여부가 영향 주지 않으나 메모리에 적재되지 않은 페이지에 액세스하려고 한다면 페이지 부재 발생



페이지 부재 처리 과정

페이지 교체(page replacement) 알고리즘

새로운 페이지가 적재되어야 하는 상황에서 빈 페이지 프레임이 없을 때 어느 페이지를 교체시킬 것인지 결정

- **FIFO(First In First Out) 알고리즘**
- **OPT (Optimal Replacement: 최적 교체) 알고리즘**
- **LRU(Least Recently Used) 알고리즘**
- **LFU(Least Frequently Used) 알고리즘**

FIFO 알고리즘

◆ FIFO 알고리즘

- 페이지가 주기억장치에 적재된(들어와 있는) 시간을 기준으로 교체될 페이지를 선정하는 기법
- 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법
- 기법
  - 주기억장치에 가장 먼저 적재된 페이지를 교체함
- 조건
  - 페이지들의 주기억장치 적재 시간을 기록해 두어야 함
- 특성
  - 이해하기 쉽고 프로그래밍 및 설계가 간단함
  - 사용 빈도가 높은 프로그램의 페이지들이 교체될 가능성 높음
- FIFO의 변칙(FIFO anomaly, 벨레이디의 모순(Belady's anomaly))
  - FIFO 알고리즘을 사용할 경우 주기억장치 할당량을 늘려 주었는데도 페이지 부재의 발생 횟수가 증가하는 경우 있음

FIFO 알고리즘

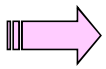
FIFO 알고리즘 사용시 페이지 부재 발생 횟수 분석

- 가정
  - 프로세스에게 4개의 페이지 프레임이 고정 할당됨
  - 초기에 4개의 페이지 프레임들이 모두 비어 있음

ω = 1 2 6 1 4 5 1 2 1 4 5 6 4 5

FIFO 기법에 의한 주기억장치 상태 변화 과정 예

시간	1	2	3	4	5	6	7	8	9	10	11	12	13	14
참조 스트링	1	2	6	1	4	5	1	2	1	4	5	6	4	5
주기억장치 상태	1	1	1	1	1	5	5	5	5	5	5	5	4	4
		2	2	2	2	2	1	1	1	1	1	1	1	5
			6	6	6	6	6	2	2	2	2	2	2	2
					4	4	4	4	4	4	4	6	6	6
페이지 부재 발생 여부	F	F	F		F	F	F	F				F	F	F



- 페이지 부재의 횟수 : 총 10번 발생

FIFO 알고리즘

FIFO Anomaly의 예 (Belady's anomaly)

◆ 가정
- 프로세스의 프로그램이 5개의 페이지로 구성되어 있음

ω = 1 2 3 4 1 2 5 1 2 3 4 5

페이지 프레임을 3개 할당했을 경우

시간	1	2	3	4	5	6	7	8	9	10	11	12
참조 스트링	1	2	3	4	1	2	5	1	2	3	4	5
주기억장치 상태	1	1	1	4	4	4	5	5	5	5	5	5
		2	2	2	1	1	1	1	1	3	3	3
			3	3	3	2	2	2	2	2	4	4
페이지 부재 발생 여부	F	F	F	F	F	F	F			F	F	

❑ 페이지 부재의 발생 횟수 : 총 9회

페이지 프레임을 4개 할당했을 경우

시간	1	2	3	4	5	6	7	8	9	10	11	12
참조 스트링	1	2	3	4	1	2	5	1	2	3	4	5
주기억장치 상태	1	1	1	1	1	1	5	5	5	5	4	4
		2	2	2	2	2	2	1	1	1	1	5
			3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	4	3	3	3
페이지 부재 발생 여부	F	F	F	F			F	F	F	F	F	F

❑ 페이지 부재의 발생 횟수 : 총 10회

OPT (Optimal Replacement: 최적 교체) 알고리즘

◆ OPT (Optimal Replacement: 최적 교체) 알고리즘

- 1966년 벨레이디(Belady)에 의해 제시된 기법
- 앞으로 가장 오랫동안 사용되지 않을 페이지를 교체시키는 것
- 페이지 부재 발생 횟수를 최소화하는 최적의 기법 (proved)
- 기법
  - 현재 시점 이후로 가장 오랫동안 참조되지 않을 페이지를 교체

- 실현 불가능한 기법
  - 각 페이지의 호출 순서와 참조 상황을 미리 예측해야 하므로 다루기가 어렵고 비현실적 (실현 가능성이 희박하다)

- 의미
  - 각 교체 기법들에 대한 성능 평가 도구로 사용 가능함

OPT (Optimal Replacement: 최적 교체) 알고리즘

OPT 알고리즘 사용시 페이지 부재 발생 횟수 분석

◆ 가정
- 프로세스에게 4개의 페이지 프레임이 고정 할당됨
- 초기에 4개의 페이지 프레임들이 모두 비어 있음

ω = 1 2 6 1 4 5 1 2 1 4 5 6 4 5

MIN 기법에 의한 주기억장치 상태 변화 과정 예

시간	1	2	3	4	5	6	7	8	9	10	11	12	13	14
참조 스트링	1	2	6	1	4	5	1	2	1	4	5	6	4	5
주기억장치 상태	1	1	1	1	1	1	1	1	1	1	1	6	6	6
		2	2	2	2	2	2	2	2	2	2	2	2	2
			6	6	6	5	5	5	5	5	5	5	5	5
					4	4	4	4	4	4	4	4	4	4
페이지 부재 발생 여부	F	F	F		F	F						F		

❑ 페이지 부재의 발생 횟수 : 총 6회

LRU(Least Recently Used) 알고리즘

◆ LRU(Least Recently Used) 알고리즘

- 참조된 시간을 기준으로 교체될 페이지를 선정하는 기법
- 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법
- 기법
  - 최근 가장 오랫동안 참조되지 않은 페이지를 교체함
  - 각 페이지마다 카운터(계수기)나 스택을 두어 현 시점에서 가장 오랫동안 사용하지 않은, 즉 가장 오래 전에 사용된 페이지를 교체함
- 조건
  - 적재되어 있는 페이지의 참조 시마다 참조 시간을 기록해야 함
- 특성
  - 프로그램의 지역성(locality)에 기반을 둠
  - 최적의 기법인 OPT 알고리즘의 성능에 근사하게 제시된 기법
- 실제로 가장 많이 사용되고 있는 기법

LRU(Least Recently Used) 알고리즘

◆ LRU(Least Recently Used) 알고리즘

- 단점
  - 매 번 참조 시간 기록해야 하는 오버헤드가 매우 큼
  - 계수기나 스택과 같은 별도의 하드웨어가 필요하며, 시간적인 오버헤드가 발생한다.
  - 실제로 구현하기가 매우 어렵다.
  - 대형 반복 구조를 실행하는데 그 보다 적은 크기의 기억장치
    공간이 할당된 경우 페이지 부재의 발생 횟수가 급격히 증가함

LRU(Least Recently Used) 알고리즘

LRU 알고리즘 사용시 페이지 부재 발생 횟수 분석

◆ 가정
- 프로세스에게 **4**개의 페이지 프레임이 고정 할당됨
- 초기에 **4**개의 페이지 프레임들이 모두 비어 있음

ω = 1 2 6 1 4 5 1 2 1 4 5 6 4 5

LRU 기법에 의한 주기억장치 상태 변화 과정 예

시간	1	2	3	4	5	6	7	8	9	10	11	12	13	14
참조 스트링	1	2	6	1	4	5	1	2	1	4	5	6	4	5
주기억장치 상태	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		2	2	2	2	5	5	5	5	5	5	5	5	5
			6	6	6	6	6	2	2	2	2	6	6	6
					4	4	4	4	4	4	4	4	4	4
페이지 부재 발생 여부	F	F	F		F	F		F				F		

❑ 페이지 부재의 발생 횟수 : 총 **7**회

LFU(Least Frequently Used) 알고리즘

◆ LFU(Least Frequently Used) 알고리즘

- 참조된 횟수를 기준으로 교체될 페이지를 선정하는 기법
- 사용 빈도가 가장 적은 페이지를 교체하는 기법
- 기법
  - 가장 참조 횟수가 적은 페이지를 교체함
  - 활발하게 사용되는 페이지는 사용 횟수가 많아 교체되지 않고 사용됨
- 조건
  - 페이지들이 참조될 때마다 참조 횟수를 누적시켜야 함
- 특징
  - LRU 기법의 오버헤드를 줄이면서 프로그램의 지역성 이용
  - 단점
    - 최근에 적재된 페이지가 이후 참조될 가능성이 많음에도 불구하고 교체될 가능성 있음
    - 각 페이지의 참조 시마다 해당 페이지의 참조 횟수를 누적시켜야 하는 오버헤드 발생
    - 프로그램 실행 초기에 많이 사용된 페이지가 그 후로 사용되지 않을 경우에도 프레임을 계속 차지할 수 있음

LFU(Least Frequently Used) 알고리즘

LFU 알고리즘 사용시 페이지 부재 발생 횟수 분석

◆ 가정
- 프로세스에게 **4**개의 페이지 프레임이 고정 할당됨
- 초기에 **4**개의 페이지 프레임들이 모두 비어 있음

ω = 1 2 6 1 4 5 1 2 1 4 5 6 4 5

LFU 기법에 의한 주기억장치 상태 변화 과정 예

시간	1	2	3	4	5	6	7	8	9	10	11	12	13	14
참조 스트링	1	2	6	1	4	5	1	2	1	4	5	6	4	5
주기억장치 상태	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		2	2	2	2	5	5	5	5	5	5	5	5	5
			6	6	6	6	6	2	2	2	2	6	6	6
					4	4	4	4	4	4	4	4	4	4
페이지 부재 발생 여부	F	F	F		F	F		F				F		

❑ 페이지 부재의 횟수 : 총 **7**번 발생

퀴즈 문제

* 다음의 페이지 참조열을 가정하자.

0, 1, 2, 3, 1, 0, 4, 5, 1, 0, 1, 2, 6, 5, 2, 1, 0, 1, 2, 5

LRU 의 페이지 교체 알고리즘에 대해 얼마나 많은 페이지 부재가 발생하는가? 4개의 페이지 프레임(page frame)이 있다고 할 때 처음에는 모든 프레임이 비어 있기 때문에 처음의 각 페이지들은 한 번씩의 페이지 부재가 발생한다.

참조열	0	1	2	3	1	0	4	5	1	0	1	2	6	5	2	1	0	1	2	5
페이지																				
프레임																				
부재																				

퀴즈 문제

* 페이지 교체 기법 중 현재 시점 이후로 가장 오랫동안 참조되지 않을 페이지를 교체시키는 것으로 페이지 부재 발생 횟수를 최소화하는 최적의 페이지 교체 기법은?

가.FIFO 나.OPT 다.LRU 라.LFU

* 프로세스에 할당된 페이지 프레임 수가 증가하면서 페이지 부재의 수가 감소하는 것이 당연하지만, 페이지 프레임수가 증가할 때 현실적으로 페이지 부재가 더 증가하는 모순(anomaly)현상과 관계 있는 페이지 교체 기법은?

가.FIFO 나.optimal

다.LRU 라.LFU

* 페이지 교체 기법 중 각 페이지마다 계수기나 스택을 두어 현 시점에서 가장 오랫동안 사용되지 않은 페이지를 교체하는 기법은?

가.FIFO 나.optimal

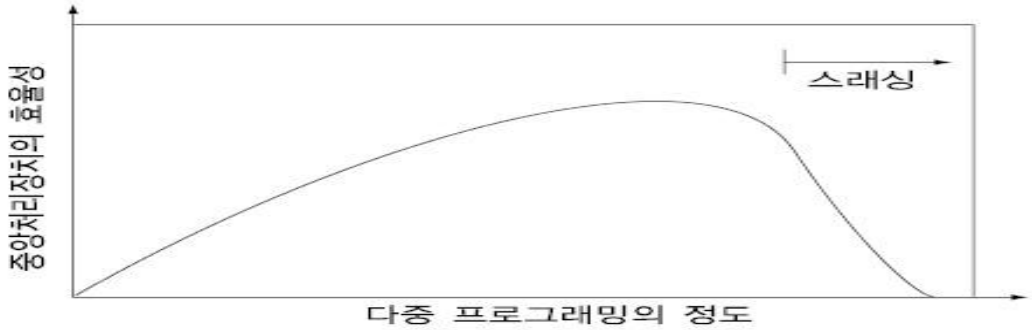
다.LRU 라.LFU

학습 내용

- 4.6 스레싱(thrashong)
- 가상 기억장치 기타 관리 사항
  - 페이지 크기
  - 지역성
  - 작업 세트
  - 페이지 부재 빈도

스래싱(thrashing)

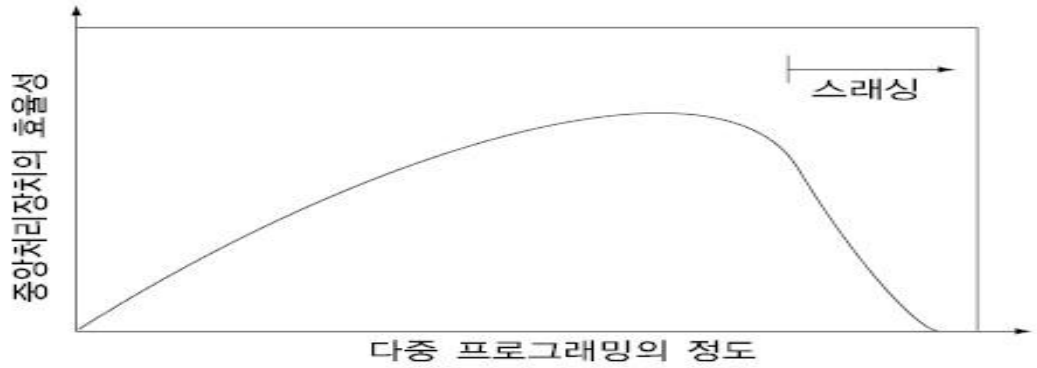
- 페이지 부재가 계속적으로 발생되어 프로세스가 수행되는 시간보다 페이지 교체에 소비되는 시간이 더 많아지는 현상
- 다중 프로그래밍 시스템이나 가상 기억장치를 사용하는 시스템에서 하나의 프로세스 수행 과정 중 자주 페이지 부재가 발생함에 따라 나타나는 현상으로 전체 시스템의 성능이 저하된다.
- 다중 프로그래밍의 정도가 높아짐에 따라 CPU의 이용률은 어느 특정 시점까지는 높아지지만, 다중 프로그래밍의 정도가 더욱 커지면 스래싱이 나타나고, CPU의 이용률은 급격히 감소하게 된다.



스래싱

스래싱(thrashing)

- 페이지 부재가 계속적으로 발생되어 프로세스가 수행되는 시간보다 페이지 교체에 소비되는 시간이 더 많아지는 현상



스래싱

- CPU의 이용률을 높이고 스래싱 현상을 방지하려면, 다중 프로그래밍의 정도를 적정 수준으로 유지하고 페이지 부재 빈도를 조절해서 사용하며, 작업 세트를 유지 해야한다.

가상 기억장치 기타 관리 사항

가상 기억장치를 구현할 때 시스템의 성능에 영향을 미치는 요소에는
페이지 크기나 지역성, 작업 세트, 페이지 부재 빈도 등이 있다.

페이지 크기

◆ 페이지 크기

- 시스템마다 다르게 결정됨
  - 시스템의 특성에 따라
  - 실행되는 프로세스들의 특성에 따라

- 일반적인 페이지 크기
  - **2^7 Bytes ~ 2^{14} Bytes**

- 장단점 비교
  - 페이지 크기를 작게 하는 경우
    - 테이블 단편화 유발로 주기억장치가 많이 소모됨
    - 페이지 프레임 개수 증가로 커널의 관리 오버헤드 발생
    - 내부 단편화를 줄일 수 있음
    - 전체적으로 디스크와 주기억장치간의 전송시간 증가
    - 사용자 프로그램 중 필요한 부분만 적재될 가능성 많아짐
  - 페이지 크기를 크게 하는 경우
    - 불필요한 부분까지 적재되어 주기억장치 공간의 낭비 초래

페이지 크기

◆ 페이지 크기

페이지 크기별 특징

작은 페이지	큰 페이지
• 페이지 테이블의 크기 증가 • 내부 단편화 감소 • 디스크 입출력 증가 • 지역성 증가와 페이지 부재 비율 증가	• 페이지 테이블의 크기 감소 • 내부 단편화 증가 • 디스크 입출력 감소 • 지역성 악화와 페이지 부재 비율 감소

지역성(locality)

- 지역성(locality)
  - 국부적인 부분만을 집중적으로 참조 한다는 것을 의미함
  - 프로그램 내의 일부 명령들이 집중적으로 실행되는 현상

  - 지역성의 원인
    - 프로그램의 반복 구조
    - 배열, 구조체 등의 자료 구조

  - 지역성의 종류
    - 시간 지역성(temporal locality)
    - 공간 지역성(spatial locality)

지역성(locality)

- 지역성(locality)의 종류
  - 시간 지역성(temporal locality)
    - 최근에 참조된 기억장소가 가까운 장래에도 계속 참조될 가능성이 높음
    - 한 번 실행된 명령은 이후 재실행될 가능성 높음
    - 한 번 접근된 데이터 영역은 이후 재접근될 가능성 높음

    예로는 순환(looping), 서브루틴, 스택, 카운팅(counting)과 집계(totaling)에 사용되는 변수들의 경우 시간구역성이 나타남

  - 공간 지역성(spatial locality)
    - 하나의 기억장소가 참조되면 그 근처의 기억장소가 계속 참조되는 경향이 있음을 의미함
    - 직전에 실행된 명령의 주변 명령들이 실행될 가능성 높음
    - 직전에 접근된 데이터의 부근 데이터들의 접근 가능성 높음

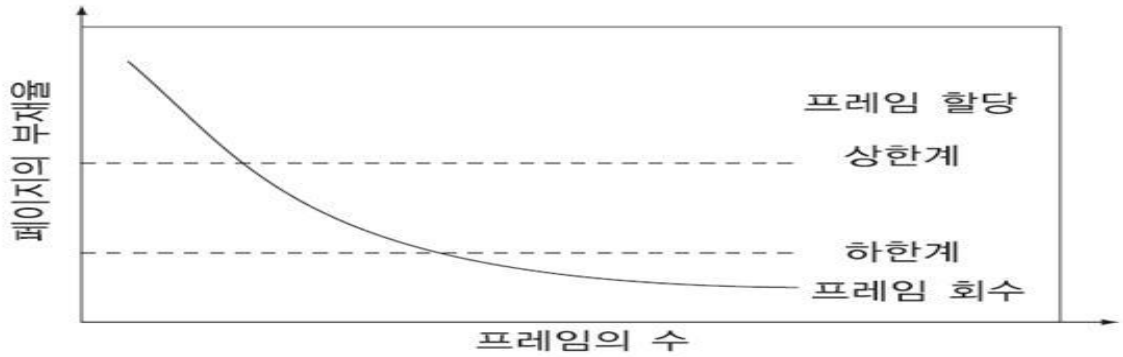
    - 예로는 배열 수행, 순차 코드의 실행(sequential code execution), 프로그래머들이 관련된 변수등

작업세트(working set)

- 작업세트(working set)
  - 프로세스에 의해 자주 참조되는 페이지들의 집합체
  - 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합
  - 운영체제가 각 프로세스의 작업 집합을 감시하고 각 프로세스에 작업 집합 크기에 맞는 충분한 프레임 할당
  - 작업 세트 방법은 가능한 다중 프로그래밍의 정도를 높이면서 스래싱을 방지하는 효과를 제공하여 프로세서의 효율성 최적화하려고 한다.
    - 자주 참조되는 작업 세트를 주기억장치에 상주시킴으로써 페이지 부재 및 페이지 교체 현상을 줄인다.
    - 빈번한 교체 과정으로 인하여 종국에는 스래싱 현상이 발생한다.

페이지 부재 빈도(Page Fault Frequency)

- 페이지 부재 빈도(페이지 부재율)
  - 페이지 부재가 일어나는 횟수



- 페이지 부재율의 조절로 스래싱 방지
- 페이지 부재 비율이 높다는 것은 프로세스에 더 많은 프레임이 필요하다는 의미이고, 페이지 부재 비율이 낮다는 것은 프로세스에 프레임이 너무 많다는 의미
- 페이지 부재율의 상한과 하한을 정해 놓고, 페이지 부재율이 상한을 넘으면 그 프로세스에게 다른 프레임을 더 할당해 줌
- 하한보다 낮으면 그 프로세스로부터 프레임을 회수 함

요약

◆ 가상기억장치의 관리 기법

- 가상기억장치 관리 기법들의 목적
  - 운영 비용의 감소
  - 비용 모델 : 일반적으로 페이지 부재의 발생 횟수로 설정됨
  - 프로그램들이 갖는 지역성 이용
- 시스템 성능에 영향을 미치는 주된 관리 기법
  - 할당 기법과 교체 기법
- 페이지(고정 할당 기반)의 교체 기법
  - 프로세스의 페이지 부재 발생 횟수를 줄이는 것이 일차적 목표
  - 실제로 많이 응용되고 있는 기법 : LRU 기법
- CPU의 이용률을 높이고 스래싱 현상을 방지
  - 다중 프로그래밍의 정도를 적정 수준으로 유지하고 페이지 부재 빈도를 조절해서 사용하며, 작업 세트를 유지해야한다.
- 가상 기억장치를 구현할 때 시스템의 성능에 영향을 미치는 요소에는 페이지 크기나 지역성, 작업 세트, 페이지 부재 빈도 등이 있다.

퀴즈 문제

* 프로그램이 실행되는 과정에서 발생하는 기억장치 참조는 하나의 순간에는 아주 지역적인 일부 영역에 대하여 집중적으로 이루어진다는 성질을 의미하는 것은?

가.Monitor 나.Thrashing 다.Locality 라,WorkingSet

* 하나의 프로세스가 자주 참조하는 페이지의 집합을 의미하며, 이런 페이지 집합니 적재되면 프로세스는 한동안 페이지 폴트 없이 실행될 수 있다. 이런 페이지 집합을 무엇이라하는가?

가. Workingset 나. Critical Section 다. paging 라. Fragmentation

* 스레싱 현상을 방지하기 위한 방법이라고 할 수 없는 것은?

가. 다중 프로그래밍의 정도를 높인다.

나. CPU이용률을 높인다.

다. 페이지 부재율을 조절하여 대치한다.

라. Workingset 방법을 사용한다.

퀴즈 문제

프로세스가 기억장치 내의 일부분만을 집중적으로 사용하는 것을 지역성이라 한다.
시간 지역성과 관련이 적은 것은?

가. Looping　나. Array Traverse　다. Stack　라. Subprogram

가상 메모리 시스템의 페이지 크기에 관한 사항으로 옳지 않은 것은?

가. 디스크 I/O 시간을 줄이기 위해서는 페이지 크기가 큰 것이 바람직하다.

나. 페이지 테이블의 크기를 고려하면 페이지 크기가 큰 것이 바람직하다.

다. 내부 단편화를 고려할 경우 페이지 크기가 큰 것이 바람직하다.

라. 페이지 크기가 커지면, 불필요한 데이터도 함께 적재될 수 있다.