가상 메모리

Virtual Memory

양희재 교수 (hjyang@ks.ac.kr) / 경성대학교 컴퓨터공학과

가상 메모리

- 물리 메모리 크기 한계 극복
  - 물리 메모리보다 큰 프로세스를 실행?
  - *e.g.* 100MB 메인 메모리에서 200MB 크기의 프로세스 실행

- 어떻게?
  - 프로세스 이미지를 모두 메모리에 올릴 필요는 없다.
  - 현재 실행에 필요한 부분만 메모리에 올린다!
  - 오류 처리 제외, 배열 일부 제외, 워드프로세스에서 정렬, 표 기능 제외 ☞ 동적 적재 (dynamic loading)과 비슷한 개념

요구 페이징

- Demand Paging
  - 프로세스 이미지는 backing store 에 저장
  - 프로세스는 페이지의 집합
  - 지금 필요한 페이지만 메모리에 올린다(load) ☞ 요구되는 (demand) 페이지만 메모리에 올린다

- 하드웨어 지원
  - *valid* 비트 추가된 페이지 테이블
  - backing store (= swap device)
  - 그림 9.8 참조

Page Fault

- 페이지 결함 (= 페이지 부재)
  - 접근하려는 페이지가 메모리에 없다 (invalid)
  - Backing store 에서 해당 페이지를 가져온다.
  - Steps in handling a page fault (그림 9.6)

- 용어: *pure demand paging* vs *prepaging*

- 비교: *swapping* vs *demand paging*

유효 접근 시간

- Effective Access Time
  - p: probability of a page fault = page fault rate
  - $T_{eff} = (1-p)T_m + pT_p$
- 예제
  - T_m = 200 nsec (DRAM)
  - T_p = 8 msec (seek time + rotational delay + transfer time)
  - $T_{eff} = (1-p)200 + p8{,}000{,}000 = 200 + 7{,}999{,}800p$
  - p = 1/1,000 ☞ T_{eff}= 8.2usec (40배 느림)
  - p = 1/399,990 ☞ T_{eff}= 220nsec (10% 느림)

지역성의 원리

- Locality of reference
  - 메모리 접근은 시간적, 공간적 지역성을 가진다!
  - 실제 페이지 부재 확률은 매우 낮다.

- 다른 방법
  - HDD 는 접근 시간이 너무 길다 ☞ swap device 로 부적합
  - SSD 또는 느린 저가 DRAM 사용

페이지 교체

Page Replacement

페이지 교체

- Demand Paging
  - 요구되어지는 페이지만 backing store 에서 가져온다.
  - 프로그램 실행 계속에 따라 요구 페이지가 늘어나고,
  - 언젠가는 메모리가 가득 차게 된다.

- Memory full!
  - 메모리가 가득 차면 추가로 페이지를 가져오기 위해
  - 어떤 페이지는 backing store 로 몰아내고 (page-out)
  - 그 빈 공간으로 페이지를 가져온다 (page-in)
  - 용어: *victim page*

Victim Page

- 어느 페이지를 몰아낼 것인가?
  - i/o 시간 절약을 위해
  - 기왕이면 modify 되지 않은 페이지를 victim 으로 선택
  - 방법: *modified bit* (= dirty bit)

- 여러 페이지 중에서 무엇을 victim 으로?
  - Random
  - First-In First-Out (FIFO)
  - 그외
  - 용어: *page replacement algorithms*

페이지 교체 알고리즘

- Page reference string
  - CPU 가 내는 주소: 100 101 102 432 612 103 104 611 612
  - Page size = 100 바이트라면
  - 페이지 번호 = 1 1 1 4 6 1 1 6 6
  - Page reference string = 1 4 6 1 6
- Page Replacement Algorithms
  - FIFO (First-In First-Out)
  - OPT (Optimal)
  - LRU (Least-Recently-Used)

First-In First-Out (FIFO)

- Simplest
  - Idea: 초기화 코드는 더 이상 사용되지 않을 것
- 예제
  - 페이지 참조열 = 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
  - # of frames = 3
  - 15 page faults
- Belady's Anomaly
  - 프레임 수 (= 메모리 용량) 증가에 PF 회수 증가?
  - 예제: Fig 9.13, p.411

Optimal (OPT)

- Rule: *Replace the page that will not be used for the longest period of time*
- 예제
  - 페이지 참조열 = 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
  - # of frames = 3
  - 9 page faults
- Unrealistic
  - 미래는 알 수 없다!
  - *cf.* SJF CPU scheduling algorithm

Least-Recently-Used (LRU)

- Rule: *Replace the page that has not been used for the longest period of time*
  - Idea: 최근에 사용되지 않으면 나중에도 사용되지 않을 것

- 예제
  - 페이지 참조열 = 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
  - # of frames = 3
  - 12 page faults

Global vs Local Replacement

- Global replacement
  - 메모리 상의 모든 프로세스 페이지에 대해 교체
- Local replacement
  - 메모리 상의 자기 프로세스 페이지에 대해 교체
- 성능 비교
  - Global replacement 가 더 효율적일 수 있다.

- 숙제!

프레임 할당

Allocation of Frames

쓰레싱 (Thrashing)

- CPU utilization vs Degree of multiprogramming
  - 프로세스 개수 증가 ☞ CPU 이용률 증가
  - 일정 범위를 넘어서면 CPU 이용률 감소
  - 이유: 빈번한 page in/out
  - Thrashing: i/o 시간 증가 때문

- 쓰레싱 극복
  - Global replacement 보다는 local replacement
  - 프로세스당 *충분한/적절한* 수의 메모리(프레임) 할당

프레임 할당

- 정적 할당 (static allocation)
  - 균등 할당 (Equal allocation)
  - 비례 할당 (Proportional allocation)
- 동적 할당 (dynamic allocation)
  - Working set model
  - Page fault frequency
  - etc.

동적 프레임 할당

- Working set model
  - Locality *vs* working set
  - Working set *window*
  - Woking set 크기 만큼의 프레임 할당

- Page-Fault Frequency (PFF)
  - Page fault 발생 비율의 상한/하한선
  - 상한선 초과 프로세스에 더 많은 프레임 할당
  - 하한선 이하 프로세스의 프레임은 회수

페이지 크기

Page Size

Page size

- 페이지 크기
  - 일반적 크기: 4KB → 4MB
  - 점차 커지는 경향
- 페이지 크기 영향
  - 내부 단편화
  - Page-in, page-out 시간
  - 페이지 테이블 크기
  - Memory resolution
  - Page fault 발생 확률

기술 동향

- 페이지 테이블
  - 원래는 별도의 chip (TLB 캐시)
  - 기술 발달에 따라 캐시 메모리는 on-chip 형태로
  - TLB 역시 on-chip 내장