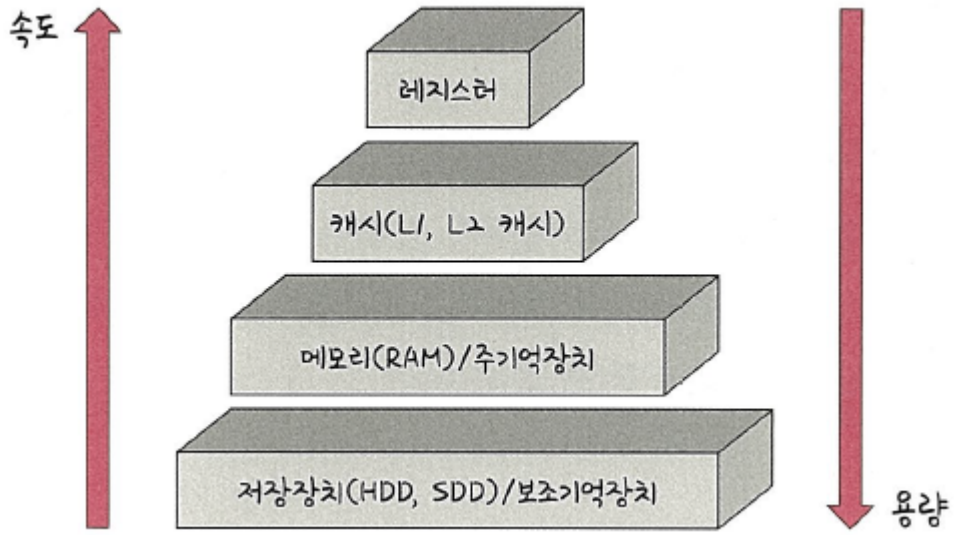
메모리

메모리 계층

레지스터, 캐시, 메모리, 저장장치로 구성되어있음



레지스터 (Register)

CPU 안에 있는 작은 메모리

휘발성 메모리

속도가 가장 빠르고, 용량이 가장 적음

캐시 (Cache)

L1, L2 캐시를 지칭함 (L3 캐시도 존재함)

휘발성 메모리

속도가 빠르고, 용량이 적음

주기억장치

RAM 을 가리킴

휘발성 메모리

일반적인 속도와 일반적인 용량

보조기억장치

HDD, SSD 를 가리킴

비휘발성 메모리

속도가 낮고, 용량이 큼

RAM 은 HDD로 부터 일정량의 데이터를 복사해서 임시 저장하고 필요시마다 CPU 에 빠르게 전달

메모리 계층의 위로 올라갈수록 가격과 속도가 증가하고, 용량이 감소함

계층은 경제성과 캐시 때문에 존재함

캐시 (Cache)

데이터를 미리 복사해놓는 임시 저장소

빠른 장치와 느린 장치에서 속도 차이에 따른 병목현상을 줄이기 위한 메모리

데이터를 접근하는 시간이 오래 걸리는 경우를 해결하고, 반복적인 계산을 하는 시간을 절약

CPU 와 메모리의 속도 차이가 매우 크므로 중간에 레지스터 계층을 두어 속도 차이를 해결함

이와 같이 속도 차이 해결을 위해 계층 사이에 존재하는 계층을 캐싱 계층이라함

지역성의 원리

캐시 계층을 두는 것이 아니라 직접 캐시를 설정해야하는 경우, 자주 사용하는 데이터를 기반으로 설정해야함

지역성은 자주 사용하는 데이터에 대한 근거가 되며, 시간 지역성 (Temporal Locality)과 공간 지역성 (Spatial Locality) 로 나뉨

시간 지역성 (Temporal Locality)

최근 사용한 데이터에 다시 접근하려는 특성

공간 지역성 (Spatial Locality)

최근 접근한 데이터를 이루고 있는 공간이나, 근접한 공간에 접근하는 특성

캐시히트, 캐시미스

캐시히트 (Cache Hit)

캐시에서 원하는 데이터를 찾은 경우

찾은 데이터를 제어장치를 거쳐 가져오게됨

위치도 가깝고, CPU 내부 버스를 기반으로 작동하므로 속도가 빠름

캐시미스 (Cache Miss)

찾는 데이터가 캐시에 없는 경우 주메모리로 가서 데이터를 찾아오는 경우

찾는 데이터를 메모리에서 가져오게 되며, 시스템 버스를 기반으로 작동하므로 속도가 느림

캐시매핑 (Cache Mapping)

캐시가 히트되기 위해 매핑하는 방법

레지스터는 RAM 에 비하면 굉장히 작으므로, 레지스터가 캐시 계층으로써 역할을 잘 수행하기 위해서는 매핑이 중요함

캐시매핑 분류

- **직접 매핑 (Directed Mapping)**

메모리가 1~100 이 있고, 캐시가 1~20 이 있는 경우, 1:1~10, 2:1~20 과 같은 방식으로 매핑하는 방식

처리가 빠르나, 충돌 발생이 잦음

- **연관 매핑 (Associative Mapping)**

  순서를 일치시키지 않고, 관련 있는 캐시와 메모리를 매핑하는 방식

  충돌이 적으나, 모든 블록을 탐색해야하므로 속도가 느림

- **집합 연관 매핑 (Set Associative Mapping)**

  직접 매핑과 연관 매핑을 합쳐 놓은 것

  순서는 일치시키나, 집합을 두어 저장함

  블록화 되어 있으므로 검색은 좀 더 효율적임

웹 브라우저의 캐시

소프트웨어적인 대표적인 캐시로는 웹 브라우저의 쿠키, 로컬 스토리지, 세션 스토리지가 있음

보통 사용자의 커스텀한 정보나 인증 모듈 관련 사항들을 웹 브라우저에 저장해서 추후 서버에 요청 시 자신을 나타내는 아이덴티티나 중복 요청 방지를 위해 사용됨

쿠키 (Cookie)

만료기한이 있는 키-값 저장소

same site 옵션은 strict 로 설정하지 않은 경우 다른 도메인에서 요청 시 자동 전송됨

4KB까지 데이터를 저장할 수 있음

만료 기한을 정할 수 있음

쿠키 설정 시에는 httponly 옵션을 걸어 document, cookie 로 쿠키를 볼 수 없게 해야함

클라이언트 또는 서버에서 만료기한 등을 정할 수 있음. 일반적으로 서버에서 정함

로컬 스토리지 (Local Storage)

만료기한이 없는 키-값 저장소

10MB까지 저장할 수 있음

웹 브라우저를 닫아도 유지되고, 도메인 단위로 저장, 생성됨

HTML5 를 지원히자 않는 브라우저는 사용 불가. 클라이언트에서만 수정이 가능

세션 스토리지 (Session Storage)

만료기한 이없는 키-값 저장소

탭 단위로 세션 스토리지를 생성함

탭을 닫을 때 해당 데이터가 삭제됨

HTML5 를 지원히자 않는 브라우저는 사용 불가. 클라이언트에서만 수정이 가능

데이터베이스의 캐싱 계층

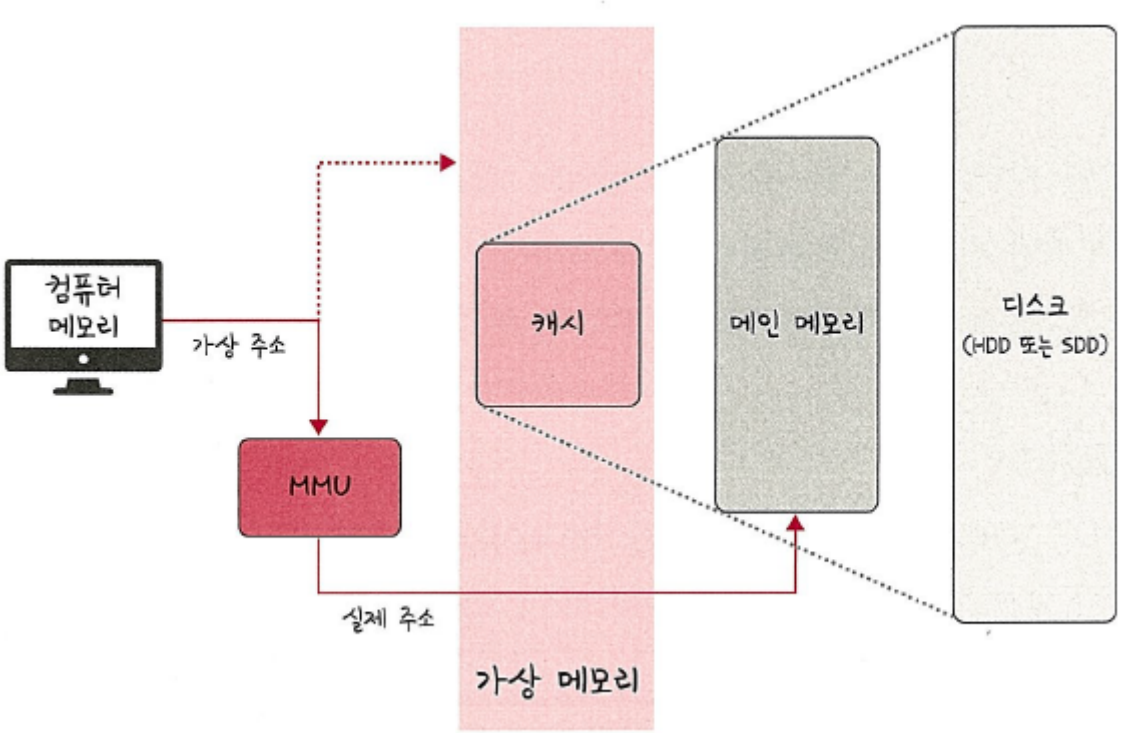
DB 시스템 구축 시에도 메인 DB 위에 레디스(redis) DB 계층을 캐싱 계층으로 두어 성능을 향상시킴

메모리 관리

가상 메모리 (Virtual Memory)

메모리 관리 기법의 하나.

컴퓨터가 실제로 이용 가능한 메모리 자원을 추상화하여 이를 사용하는 사용자들에게 매우 큰 메모리로 보이게 만드는 것



가상 주소 (Logical Address) : 가상적으로 주어진 주소

실제 주소 (Physical Addresss) : 실제 메모리 상에 있는 주소

가상주소는 메모리 관리 장치 (MMU) 에 의해 실제 주소로 변환됨 → 사용자는 실제 주소를 의식할 필요 없이 프로그램 구축을 할 수 있음

가상메모리는 가상 주소와 실제 주소가 매핑 되어 있고, 프로세스의 주소 정보가 들어있는 페이지 테이블로 관리되며, 이때 속도 향상을 위해 TLB 를 사용함

TLB

메모리와 CPU 사이에 있는 주소 변환을 위한 캐시

페이지 테이블에 있는 리스트를 보관하며 CPU가 페이지 테이블까지 가지 않도록 하여 속도를 향상 시킬 수 있는 캐시 계층

스와핑 (Swapping)

가상 메모리에는 존재하지만, 실제 메모리인 RAM에는 현재 없는 데이터나 코드에 접근할 경우 페이지 폴트 (Page Fault)가 발생함

이를 방지하기 위해 당장 사용하지 않는 영역을 HDD로 옮겨 필요 시 다시 RAM 으로 불러오고, 사용하지 않으면 다시 HDD 로 옮기는 것을 반복하여 RAM 을 효과적으로 관리하는 것을 스와핑이라고함

페이지 폴트 (Page Fault)

프로세스의 주소 공간에는 존재하나 RAM 에는 없는 데이터에 접근했을 경우 발생

이때 OS는 다음 과정으로 해당 데이터를 메모리로 가져와서 페이지 폴트가 발생하지 않은 것처럼 프로그램이 작동하게함.

페이지 폴트와 그로 인한 스와핑은 아래의 과정으로 이루어짐

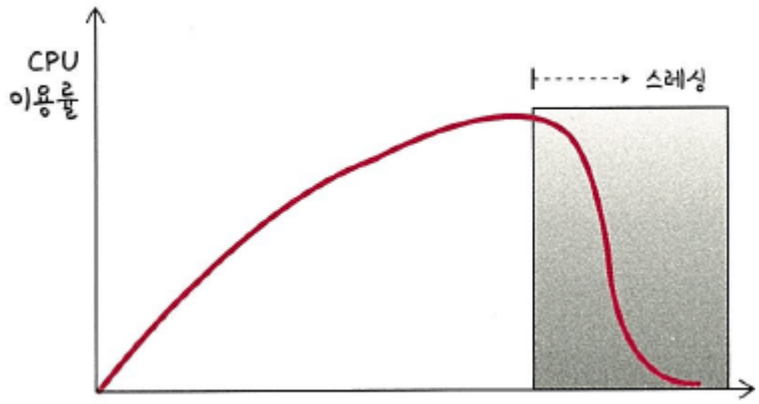
1. CPU는 물리 메모리를 확인하여 해당 페이지가 없으면 트랩을 발생해서 OS에 알림
2. OS는 CPU의 동작을 잠시 멈춤
3. OS는 페이지 테이블을 확인하여 가상 메모리에 페이지가 존재하는지 확인하고, 없으면 프로세스를 중단하고 현재 물리 메모리에 비어 있는 프레임이 있는지 찾음. 물리 메모리에도 없다면 스와핑이 수행됨
4. 비어 있는 프레임에 해당 페이지를 로드하고, 페이지 테이블을 갱신함
5. 중단되었던 CPU를 다시 시작함

페이지 (Page) : 가상 메모리를 사용하는 최소 크기 단위

프레임 (Frame) : 실제 메모리를 사용하는 최소 크기 단위

스레싱 (Thrashing)

메모리의 페이지 폴트율이 높은 것. 이는 컴퓨터의 심각한 성능 저하를 초래함



원인

메모리에 너무 많은 프로세스가 동시에 올라가게 되면 스와핑이 많이 일어남

페이지 폴트가 일어나면 CPU 이용률이 낮아지고, CPU 이용률이 낮아지면

→ OS 는 CPU 의 작업이 없는 것으로 판단하여 가용성을 더 높이기 위해 더 많은 프로세스를 메모리에 올리게 됨 → 악순환 반복

해결

메모리 증가

HDD 를 사용하는 경우, SSD로 교체

OS 에서 해결할 수 있는 방법은 작업 세트와 PFF 가 있음

작업 세트 (Working Set)

프로세스의 과거 사용 이력인 지역성(locality)을 통해 결정된 페이지 집합을 만들어서 미리 메모리에 로드하는 것

미리 메모리에 로드하면 탐색에 드는 비용을 줄일 수 있고 스와핑 또한 줄일 수 있음

PFF (Page Fault Frequency)

상한선과 하한선을 만들어 페이지 폴트 빈도를 조절하는 방법

상한선 도달 시 페이지를 늘리고 하한선 도달 시 페이지를 줄임

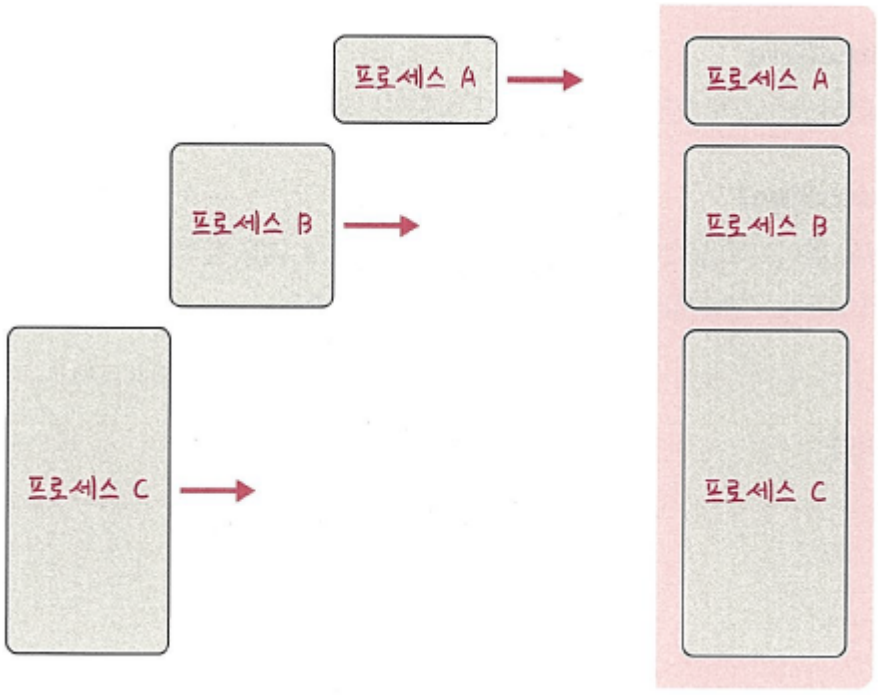
메모리 할당

메모리에 프로그램을 할당할 때는 시작 메모리 위치, 메모리의 할당 크기를 기반으로 할당

연속 할당과 불연속 할당으로 나뉨

연속 할당

메모리에 연속적으로 공간을 할당하는 것



위 그림과 같이 프로세스 A, B, C 가 순차적으로 공간에 할당됨

이는 메모리를 미리 나누어 관리하는 고정 분할방식과 매 시점 프로그램의 크기에 맞게 메모리를 분할하여 사용하는 가변 분할 방식이 있음

고정 분할방식 (Fixed Partition Allocation)

메모리를 미리 나누어 관리하는 방식

메모리가 미리 나뉘어 있으므로 융통성이 없음

`내부 단편화 (Internal Fragmentation)` 가 발생함

- 메모리를 나눈 크기보다 프로그램이 작아서 들어가지 못하는 공간이 많이 발생하는 현상

가변분할방식 (Variable Partition Allocation)

매 시점 프로그램의 크기에 맞게 동적으로 메모리를 나누어 사용함

`외부 단편화 (External Fragmentation)` 가 발생할 수 있음

- 메모리를 나눈 크기보다 프로그램이 커서 들어가지 못하는 공간이 많이 발생하는 현상
  - ex) 100MB 를 55, 45MB 로 나눴는데 프로그램의 크기가 70MB 인 경우

최초 적합, 최적적합, 최약적합의 3가지가 있음

종류	설명
최초 적합 (First Fit)	위쪽이나 아래쪽부터 시작해서 홀을 찾으면 바로 할당
최적 적합 (Best Fit)	프로세스의 크기 이상인 공간 중 가장 작은 홀부터 할당
최악 적합 (Worst Fit)	프로세스의 크기와 가장 많이 차이가 나는 홀에 할당

- **홀 (Hole) :** 할당할 수 있는 비어있는 메모리 공간

불연속 할당

메모리를 연속적으로 할당하지 않는 방식

현대 OS 가 쓰는 방법으로 페이징 기법이 있으며, 이외에도 세그멘테이션, 페이지드 세그멘테이션이 있음

페이징 (Paging)

메모리를 동일한 크기의 페이지 (일반적으로 4KB)로 나누고 프로그램마다 페이지 테이블을 두어 이를 통해 메모리에 프로그램을 할당하는 기법

동일한 크기의 페이지 단위로 나누어 메모리의 서로 다른 위치에 프로세스를 할당하여 홀의 크기가 균일하지 않은 문제는 해결하나, 주소 변환이 복잡해짐

세그멘테이션 (Segmentation)

페이지 단위가 아닌 의미 단위인 세그먼트 (Segment) 로 나누는 방식

프로세스는 코드, 데이터, 스택, 힙 등으로 이루어지는데, 코드와 데이터 등 이를 기반으로 나눌 수도 있으며, 함수 단위로도 나눌 수 있음

공유와 보안 측면에서 장점을 가지나, 홀 크기가 균일하지 않은 문제가 발생

페이지드 세그멘테이션 (Paged Segmentation)

공유나 보안을 의미 단위의 세그먼트로 나누고, 물리적 메모리는 페이지로 나누는 것

페이지 교체 알고리즘

메모리는 한정되어있음 → 스와핑이 많이 일어남

그러나 스와핑은 많이 일어나지 않도록 설계되어야하며, 페이지 교체 알고리즘을 기반으로 스와핑이 일어남

오프라인 알고리즘 (Offline Algorithm)

먼 미래에 참조되는 페이지와 현대 페이지를 바꾸는 알고리즘. 가장 좋은 방법

그러나, 미래에 사용되는 프로세스를 알 방법이 없음. 즉, 사용할 수 없는 알고리즘

`다른 알고리즘과의 성능 비교에 대한 기준을 제공하는 알고리즘`

FIFO (First In First Out)

가장 먼저 온 페이지를 교체 영역에 가장 먼저 놓는 방법

LRU (Least Recently Used)

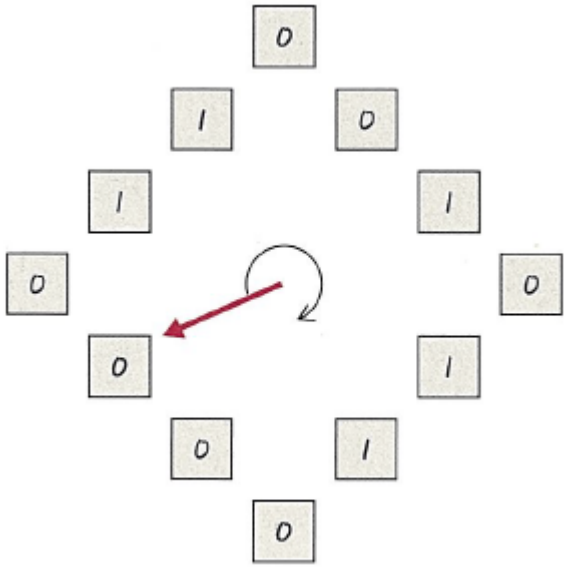
참조가 가장 오래된 페이지를 바꾸는 방식

오래된 것을 파악하기 위해 각 페이지마다 계수기, 스택을 두어야하는 문제점이 있음

프로그래밍으로 구현 시 일반적으로 두개의 자료구조로 구현함

- 해시 테이블 : 이중 연결 리스트에서의 탐색을 빠르게 하기 위해 사용
- 이중 연결 리스트 : 한정된 메모리를 나타냄

NUR (Not Used Recently)



LRU 에서 발전한 알고리즘

일명 Clock 알고리즘이라고 함

0 과 1을 가진 비트를 둠 (1은 최근에 참조되었고, 0은 참조되지 않았음을 의미)

시계 방향으로 돌면서 0을 찾고, 찾은 순간 해당 프로세스를 교체하고, 해당 부분을 1로 바꿈

LFU (Least Frequently Used)

가장 참조 횟수가 적은 페이지를 교체