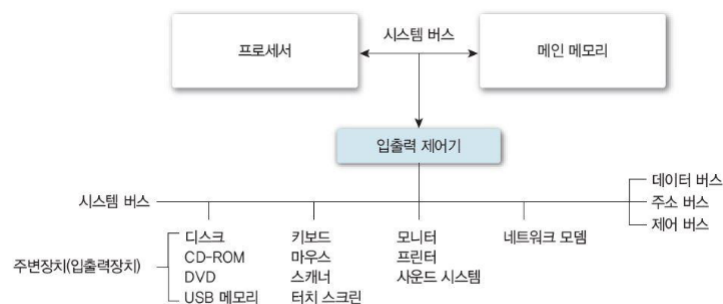


12. Disk Management

Overview

- I/O mechanisms
 - How to send data between processor and I/O device
- I/O services of OS
 - OS supports for better I/O performance
- Disk scheduling
 - Improve throughput of a disk
- RAID architecture
 - Improve the performance and reliability of disk system

I/O System (HW)



- I/O
 - 프로세서가 필요한 정보를 요청하면 데이터를 읽어오거나 내보내는 것
- 입출력 제어기
 - 입출력 장치로부터 어떤 데이터를 주거나 입출력 장치에 어떤 데이터를 보내는 역할
- 메인 메모리
 - 입출력 장치 ↔ 프로세서 간 왔다갔다하는 데이터를 올려놓음
 - 필요한 데이터는 무조건 메인 메모리를 거쳐 사용하게 됨

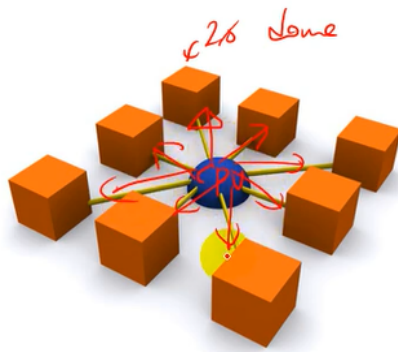
I/O Mechanisms

| 프로세서와 I/O device 간 데이터를 보내는 방법

- Processor controlled memory access
 - Polling (programmed I/O)
 - Interrupt
- Direct Memory Access (DMA)

Processor controlled memory access

Polling (Programmed I/O)



- **Processor가 주기적으로 I/O 상태 확인**하면서 줄/받을 데이터가 있으면 처리함
 - 모든 I/O 장치를 순환하며 확인
 - ex. 전송 준비, 전송 상태 등
- **장점**
 - 단순함
 - I/O 장치가 빠르고, 데이터 전송이 잦은 경우 효율적
- **단점**
 - 프로세서 부담이 큼: 계속 확인하고 있어야 함
 - Polling Overhead 발생: I/O device가 느린 경우

Interrupt



[그림출처] 운영체제, 한빛미디어

- I/O 장치가 작업을 완료한 후, 자신의 상태를 프로세서에게 전달
 - Interrupt 발생 시, 프로세서는 데이터 전송 수행
- 장점
 - Polling 대비 낮은 오버헤드
 - 불규칙적인 요청 처리에 적합
- 단점
 - Interrupt handling overhead: 인터럽트가 자주 발생하는 경우
- ex. 윈도우의 장치 관리자에서 인터럽트 요청 발생한 것을 확인할 수 있음

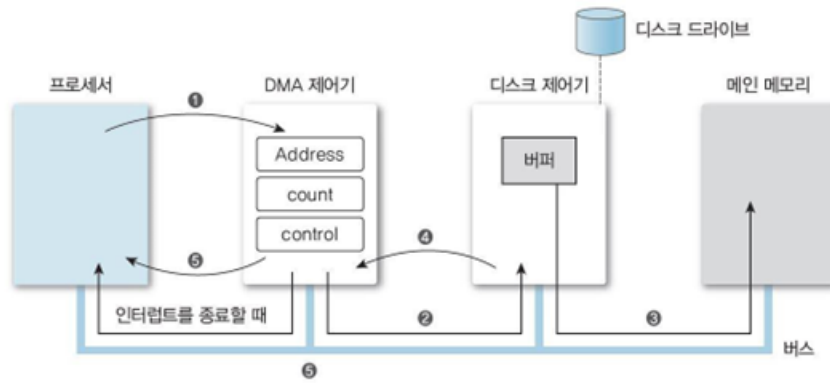
Direct Memory Access (DMA)

- Processor Controlled Memory Access 방식의 단점
 - 프로세서가 매번 모든 데이터 전송을 처리해야 함 ⇒ 프로세서에 높은 오버헤드 발생
- Direct Memory Access



중간에 DMA를 넣음으로써 구현이 가능하게 됨

- I/O 장치와 메모리 사이의 데이터 전송을 프로세서 개입 없이 수행
- 프로세서는 데이터 전송의 시작/종료에만 관여



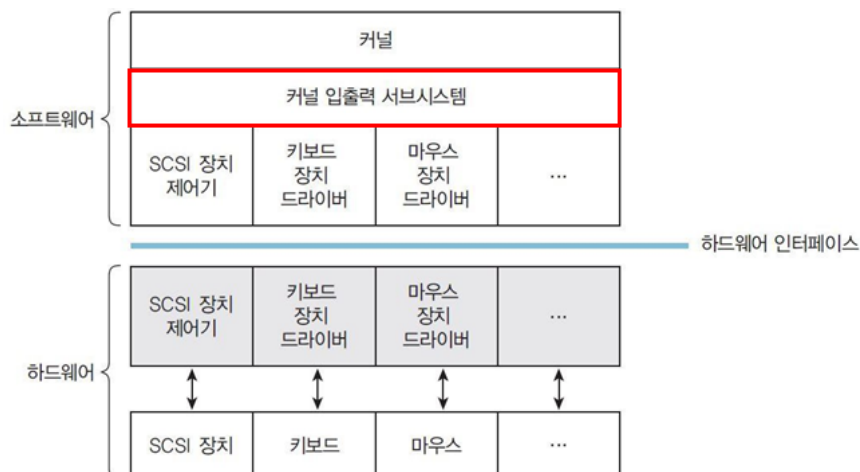
- ① 프로세서가 전송 방향, 전송 바이트 수, 데이터 블록의 메모리 주소 등을 DMA 제어기에 보낸다.
- ② DMA 제어기는 디스크 제어기에 데이터를 메인 메모리로 전송하라고 요청한다.
- ③ 디스크 제어기가 메인 메모리에 데이터를 전송한다.
- ④ 데이터 전송을 완료하면 디스크 제어기는 DMA 제어기에 완료 메시지를 전달한다.
- ⑤ DMA 제어기가 프로세서에 인터럽트 신호를 보낸다.

메인 메모리에 데이터를 보내는 경우

1. 프로세서가 DMA 제어기에 어떤 작업, 어떤 데이터, 어떤 장치에게 보내야할지 등 보냄
2. DMA 제어기가 디스크 제어기에 데이터를 메인 메모리로 전송하라고 요청
3. 디스크 제어기가 메인 메모리에 데이터 전송
4. 디스크 제어기가 전송 완료 후, DMA 제어기에 완료 메시지 전송
5. DMA 제어기가 프로세서에 Interrupt 신호로 끝났음을 알림

I/O Services of OS

I/O 성능을 높이기 위해 OS에서 할 수 있는 일



커널 내 커널 입출력 서브시스템이 I/O 서비스 제공

I/O Scheduling

- 입출력 요청에 대한 처리 순서 결정
 - 시스템의 전반적 성능 향상
 - 프로세스의 요구에 대한 공평한 처리
- ex. Disk I/O scheduling: 여러 파일을 복사할 때, 복사 순서 등

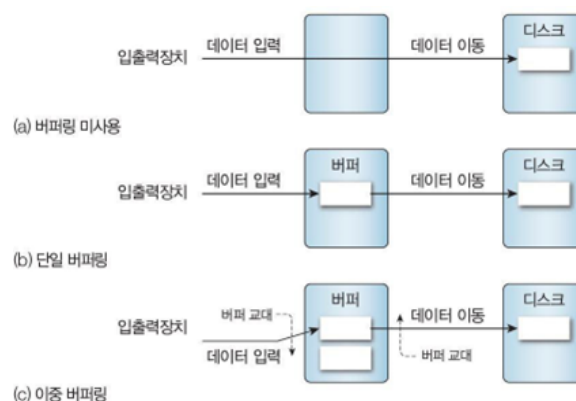
Error Handling

- 입출력 중 발생하는 오류 처리
- ex. disk access fail, NW communication error 등

I/O device information managements

- 시스템의 다양한 I/O 장치들에 대한 정보 관리

Buffering



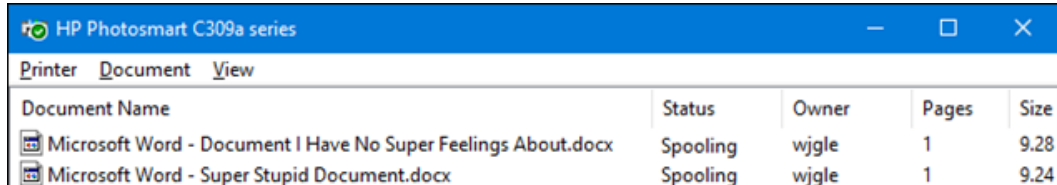
- I/O 장치와 프로그램 사이에 전송되는 데이터를 buffer에 임시 저장
 - ex. 동영상 재생 전 버퍼링
- 전송 속도 (or 처리 단위) 차이 문제 해결
 - ex. 입력과 출력 속도 차이. 입력 속도가 출력 속도보다 빠르다면, 버퍼에 미리 입력 해두고 디스크가 처리할 수 있는 만큼만 처리하도록 함

Caching

- 자주 사용하는 데이터를 미리 복사해 둬

- Cache hit 시 I/O 생략할 수 있음
 - 메모리 공간 상 일정 공간을 잡아두고 자주 사용하는 데이터를 저장해둬. 프로세서 (CPU)는 입출력 장치까지 요청할 필요 없이 메인 메모리까지만 가면됨

Spooling



Document Name	Status	Owner	Pages	Size
Microsoft Word - Document I Have No Super Feelings About.docx	Spooling	wjgle	1	9.28
Microsoft Word - Super Stupid Document.docx	Spooling	wjgle	1	9.24

- 한 I/O 장치에 여러 프로그램이 요청을 보낼 시, 출력이 섞이지 않도록 하는 기법
 - 각 프로그램에 대응하는 disk file에 기록 (spooling)
 - Spooling이 완료되면, spool을 한 번에 하나씩 I/O 장치로 전송

Disk Scheduling

- 정의
 - Disk access 요청들의 처리 순서를 결정
 - Disk system의 성능을 향상
- 평가 기준
 1. **Throughput**
 - 단위 시간당 처리량
 2. **Mean response time**
 - 평균 응답 시간: 데이터 요청에서 전달까지 시간
 3. **Predictability**
 - 응답 시간의 예측성. 무한 대기 X
 - 요청이 무기한 연기(starvation)되지 않도록 방지

Data Access Time

$$\text{Data Access Time} = 1+2+3$$

- 최적화할 수 있는 것은 1과 2

1. Seek Time (탐색 시간)

- 디스크 head를 필요한 cylinder로 이동하는 시간

2. Rotational Delay

- 1) Seek Time 이후로부터, 필요한 sector가 head 위치로 도착하는 시간

3. Data Transmission Time

- 2) Rotational Delay 이후로부터, 해당 sector를 읽어서 전송(or 기록)하는 시간

Optimizing Seek Time

First Come First Service (FCFS)

요청이 도착한 순서에 따라 처리

• 장점

- 간단함 ➡ 스케줄링 오버헤드 낮음
- 공평한 처리 기법 (무한 대기 방지)

• 단점

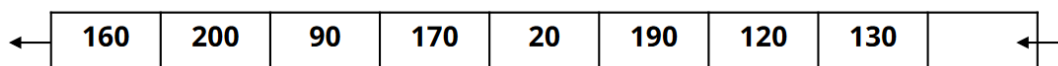
- 최적 성능 달성에 대한 고려 없음. 들어온 순서대로 처리

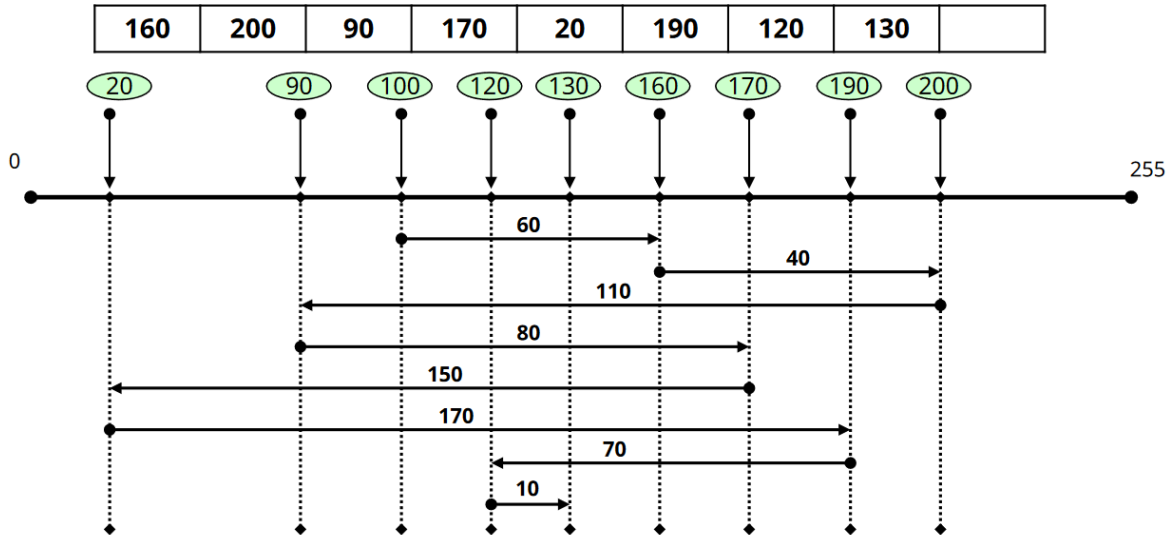
- **Disk access 부하가 적은 경우에 적합** (Disk access 적은 경우)

[예시]

• 조건

- 총 256개의 cylinder로 구성
- Head의 시작 위치: 100번 cylinder
- Access request queue





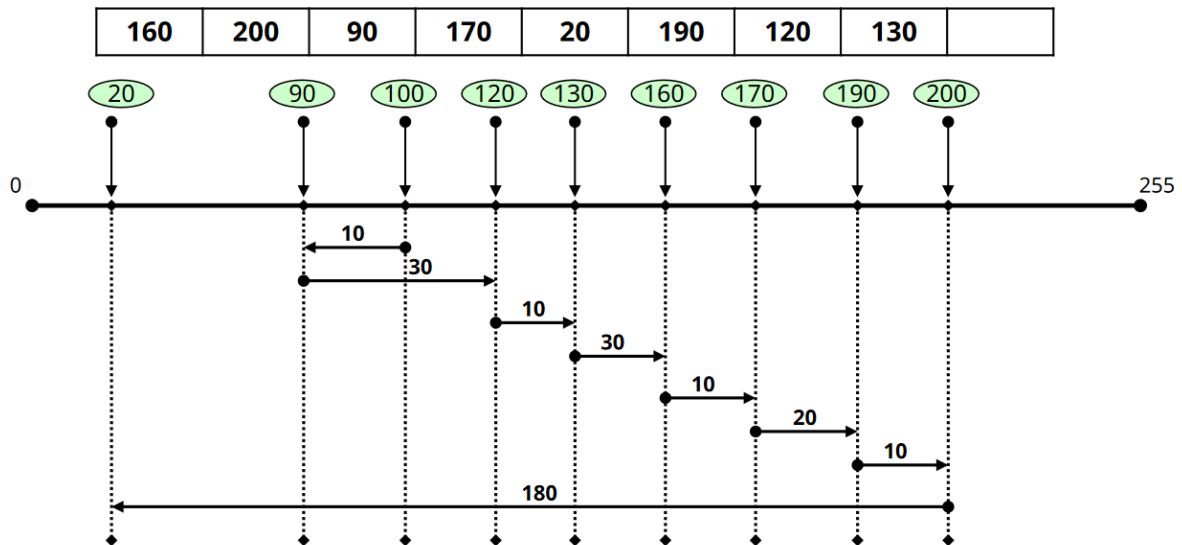
- 총 이동 거리(Total seek distance) = 690

Shortest Seek Time First (SSTF)

현재 head 위치에서 가장 가까운 요청 먼저 처리

- 이동 거리 줄어듦
- 장점
 - Throughput ↑
 - 평균 응답 시간 ↓
- 단점
 - Predictability ↓
 - 멀리 떨어져 있는 경우, 언제 처리 받을지 모름
 - Starvation 현상 발생 가능
 - 계속 현재 head 근접한 요청이 지속적으로 들어온다면?
- 일괄 처리 시스템에 적합
 - Throughput 중요, 평균 응답 시간 중요 X

[예시]



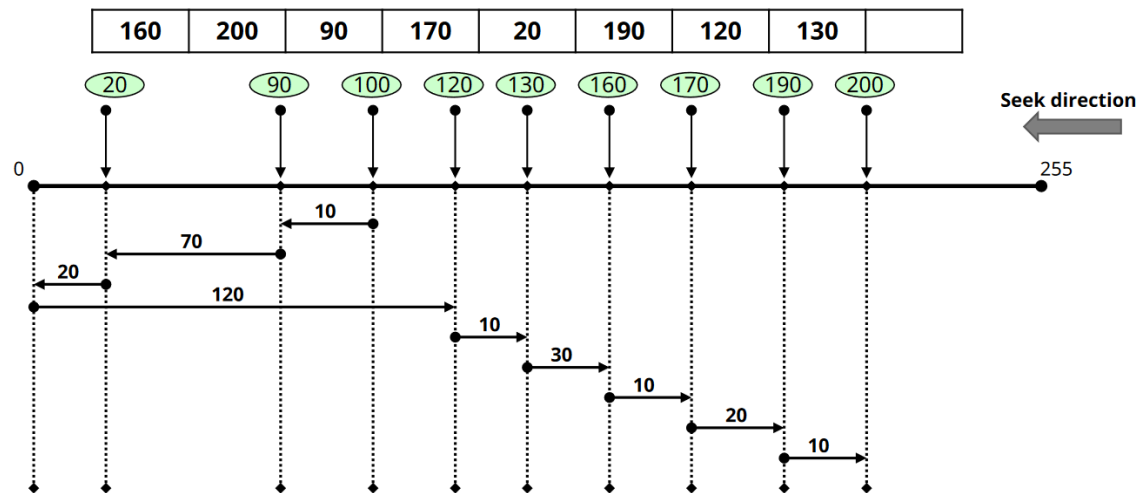
- Total seek distance = 300
 - 20에 대해서 매우 오래 걸렸다는 것, starvation 발생 가능이라는 걸 알 수 있음

Scan Scheduling

현재 head의 진행 방향에서 head와 가장 가까운 요청 먼저 처리
(진행 방향 기준) 마지막 cylinder 도착 후, 반대 방향으로 진행

- 장점
 - SSTF의 starvation 문제 해결
 - Throughput 및 평균 응답시간 우수
- 단점
 - 진행 방향 반대쪽 끝의 요청들의 응답시간 ↑

[예시]



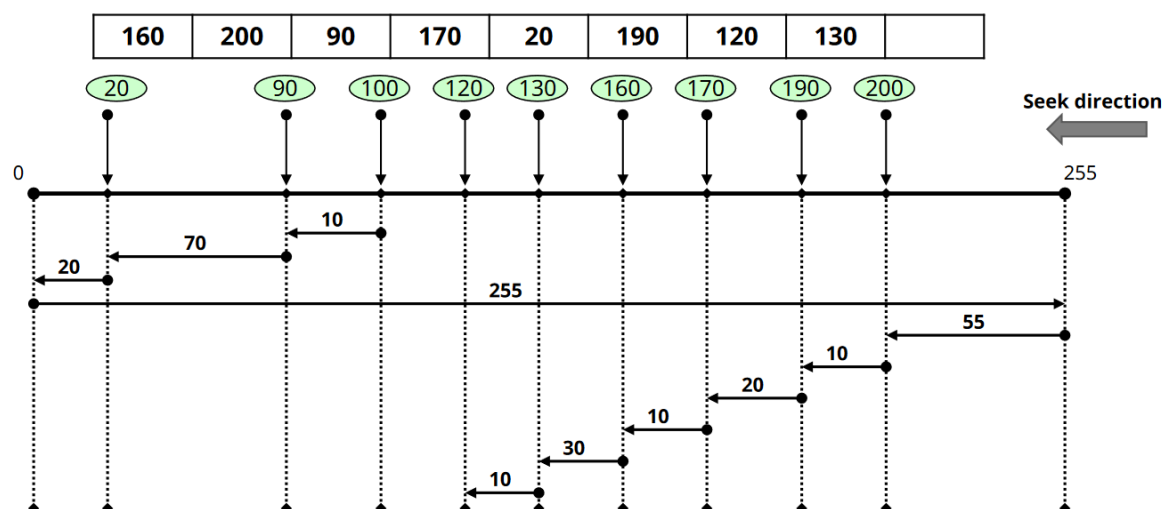
- Total seek distance = 300
 - SSTF와 같음. starvation 문제 발생 X

C-Scan Scheduling

Scan과 유사. Head가 미리 정해진 방향으로만 이동

- 마지막 cylinder 도착 후, 시작 cylinder로 이동 후 재시작
- 장점
 - Scan 대비 균등한 기회 제공

[예시]



- Total seek distance = 490

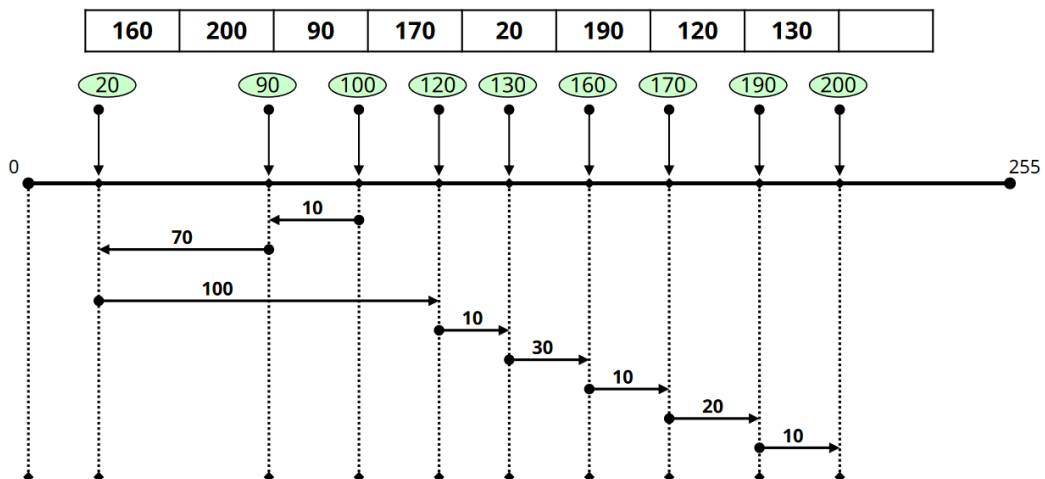
- Scan과 달리 전체적으로 기회가 균등하게 돌아감
- 중간에 불필요한 횡단이 발생함

Look Scheduling

aka Elevator Algorithm. Scan(C-scan)에서 현재 진행 방향에 요청이 없으면 방향 전환

- 마지막 cylinder까지 이동 X (요청한 곳까지만 왔다가 이동)
- Scan (C-Scan)의 실제 구현 방법
- 장점
 - Scan의 불필요한 이동 제거

[예시]



- Total seek distance = 260

Optimizing Rotational Delay

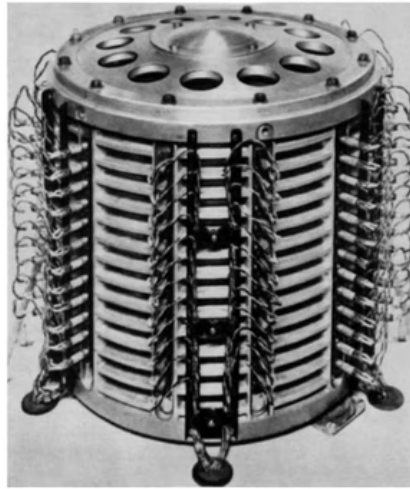
Shortest Latency Time First (SLTF)

회전수를 최대한 줄이려는 방식

- Fixed head disk 시스템 사용

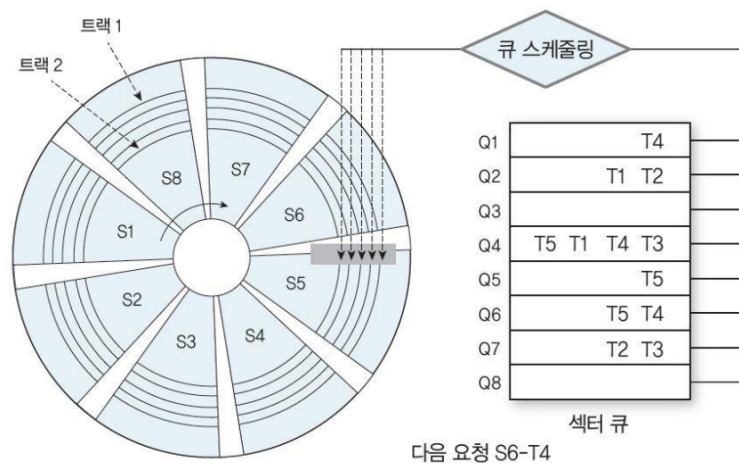
- 각 track마다 head를 가진 disk. head 이동 X

▼ ex. drum disk



Drum memory of
a Polish ZAM-41 computer

[Sector queuing algorithm]



- 각 sector별 queue 유지
- Head 아래 도착한 sector의 queue에 있는 요청을 먼저 처리한 수, head 이동
- **Moving head disk의 경우**
 - 같은 cylinder 또는 track에 여러 개의 요청 처리를 위해 사용 가능
 - Head가 특정 cylinder에 도착하면 고정 후, 해당 cylinder의 요청 모두 처리

Shortest Positioning Time First (SPTF)

Positioning time이 가장 작은 요청 먼저 처리

- Positioning time = Seek time + Rotational delay

- 장점

- Throughput ↑, 평균 응답 시간 ↓

- 단점

- 가장 안쪽과 바깥쪽 cylinder의 요청에 대해 starvation 현상 발생 가능
 - 계속 근접한 위치만 이동하게 될 수도 있음

[Eschenbach Scheduling]

- Positioning time 최적화 시도 (SPTF 개선)
- Disk가 1회전 하는 동안 요청을 처리할 수 있도록 요청을 정렬
 - 한 cylinder 내 track, sector들에 대한 다수의 요청이 있는 경우, 다음 회전에 처리
- 단점
 - 한 cylinder에만 요청이 모여있다면, 비효율적일 수 있음

RAID Architecture

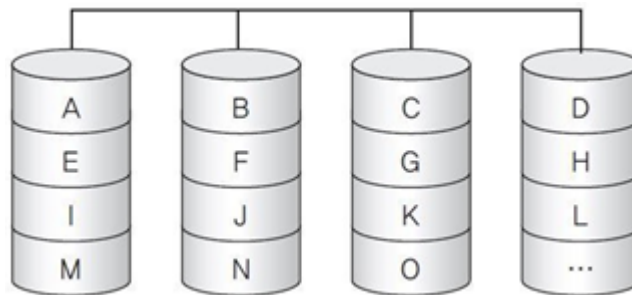
Redundant Array of Inexpensive Disks (RAID)

- 여러 개의 물리 disk를 하나의 논리 disk로 사용
 - OS support, RAID controller
- Disk system의 성능 향상을 위해 사용
 - Performance(access speed): 얼마나 빨리 데이터를 가져오는가
 - Reliability: 데이터 신뢰성

RAID 0

Disk striping ⇒ Performance ↑

- 논리적인 한 block(디스크 내부 단위)을 일정한 크기로 나눠 각 disk에 나누어 저장



• 장점

- 모든 disk에 입출력 부하 균등 분배
 - Parallel access로 성능 향상
 - Performance 향상
 - 아주 이상적인 경우, disk 4개 쓰면 4배 성능 향상

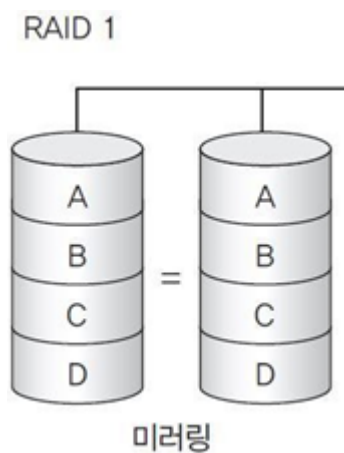
• 단점

- 한 Disk에서 장애 시, 데이터 손실 발생
 - Low reliability

RAID 1

Disk mirroring ⇒ Reliability ↑

- 동일한 데이터를 mirroring disk에 중복 저장

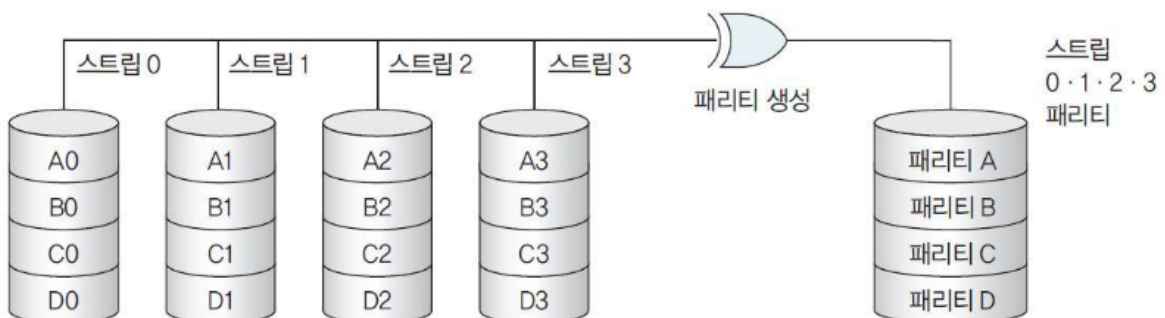


- 최소 2개의 disk로 구성
 - 입출력은 둘 중 어느 disk에서도 가능
- 장점
 - 한 disk에 장애가 생겨도 데이터 손실 X
 - High reliability
- 단점
 - 가용 disk 용량 = (전체 disk 용량/2)

RAID 3

RAID 0 + parity disk (데이터 복구할 수 있는 정보)

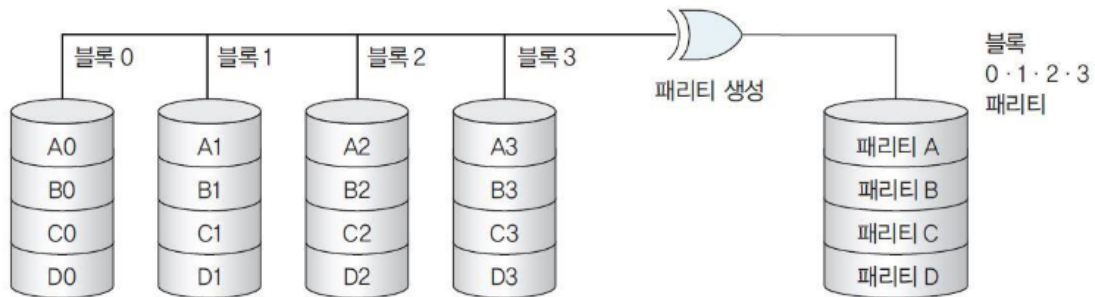
- Byte 단위 분할 저장, 스트립별 나눠 저장
- 모든 disk에 입출력 부하 균등 분배



- 한 disk에 장애 발생 시, parity 정보를 이용하여 복구
- Write 시 parity 재계산 필요
 - Overhead 발생
 - Write가 몰릴 시, 병목현상 발생 가능
- RAID 0의 신뢰성 문제 개선(parity bit)

RAID 4

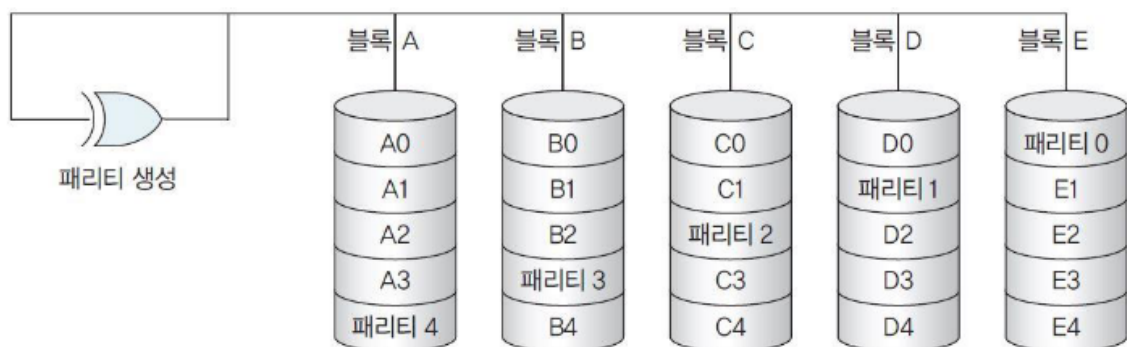
RAID 3과 유사. 단 block 단위로 분산 저장



- 각 블록을 독립적으로 access 가능
 - RAID 3은 A0~A3 다 읽어와야 했지만, RAID 4는 A0만 읽어올 수 있음
- Disk 간 균등 분배가 안될 수 있음
 - A0~D0만 필요하다면, 그쪽에만 몰릴 수도
 - 한 disk에 입출력이 몰릴 때, 병목 현상으로 성능 저하 가능
- 한 disk에 장애 발생 시, parity 정보를 이용하여 복구
 - Write 시 parity 계산 필요 \Rightarrow Overhead/Write가 몰릴 시 병목 현상 발생 가능

RAID 5

RAID 4와 유사. Parity 정보를 각 disk들에 분산 저장



- 독립된 access 방법 (RAID 4와 동일)
- Parity 정보를 각 disk들에 분산 저장
 - Parity disk의 병목 현상 문제 해소

- RAID 4와 달리 Parity 정보가 분산 저장되어, reliability ↑
- **현재 가장 널리 사용되는 RAID level 중 하나**
 - High Performance & Reliability