Slide 0

4부 입출력 시스템
(14장. 대용량 저장장치 구조)

디스크 구조

Slide 1

- 소개
  - 디스크는 현대 컴퓨터 시스템의 대용량 보조저장장치로 사용
  - 데이터 전송은 논리 블록 단위로 전송
  - 논리적 블록의 크기는 보통 512B이나 1024B등도 가능
  - 디스크 구동기는 논리 블록을 1차원 배열의 주소로 다룸
  - 논리블록은 순차적인 디스크 섹터로 매핑됨
  - 섹터 0은 최 외곽 실린더의 첫번째 트랙의 첫번째 섹터
  - 논리 블록 번호 →실린더 번호, 트랙번호, 섹터번호로 변환됨

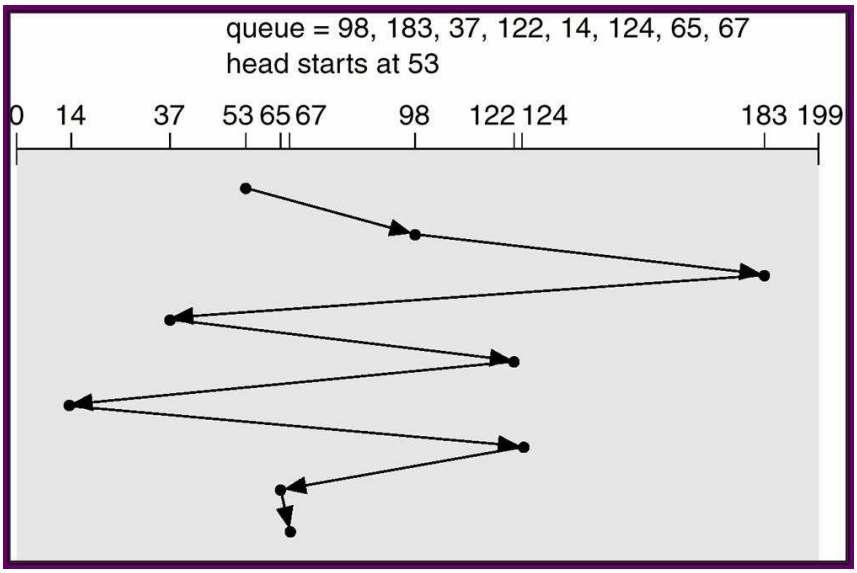
디스크 스케줄링

Slide 2

- 디스크 스케줄링이란?
  - 프로세스들의 디스크 요청이 다량으로 대기 중일때
  - 이 요청의 순서를 바꾸어 디스크 입출력을 효율적으로 하는 작업
- 선입선출 (FCFS) 스케줄링
  - 가장 단순한 방법
  - 먼저 온것을 먼저 처리
  - 디스크 요청이 적을 경우에 효과적인 방법
  - 예에서 총 640 헤드 이동이 있음

디스크 스케줄링 (계속)

Slide 3



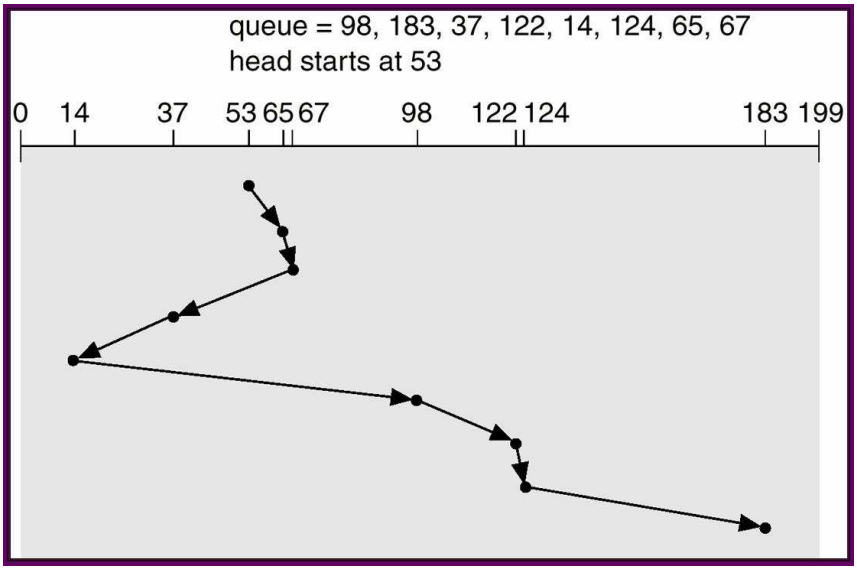
디스크 스케줄링 (계속)

Slide 4

- 최소 탐색 우선(SSTF) 스케줄링
  - 디스크 헤드에 가까운 것을 먼저 처리
  - 탐색시간이 가장 적게 드는 것을 먼저 처리
  - 헤드 이동 236으로 FCFS 에 비해 1/3 정도임
  - 이론적으로 무한적으로 기다리는 요청이 있을 수 있음
    (현재 헤드 위치에 가까운 요청이 계속 들어올 경우)

디스크 스케줄링 (계속)

Slide 5



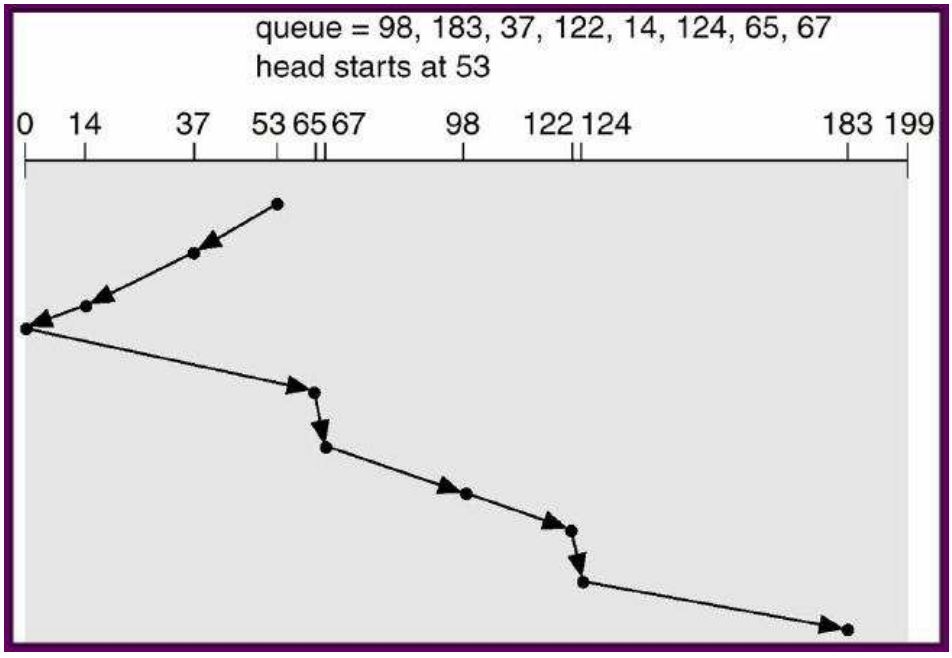
디스크 스케줄링 (계속)

Slide 6

- SCAN 스케줄링
  - 한 방향을 정해서 그 쪽 방향에 것을 먼저 처리 후 끝에 다달으면 방향 전환
  - 헤드는 양 끝을 왔다 갔다함
  - 방향의 바로 뒷쪽의 요구는 거의 한 번을 순회할 때까지 기다려야함
  - 엘리베이터와 동작이 유사하여 엘리베이터 알고리즘으로도 불림

디스크 스케줄링 (계속)

Slide 7



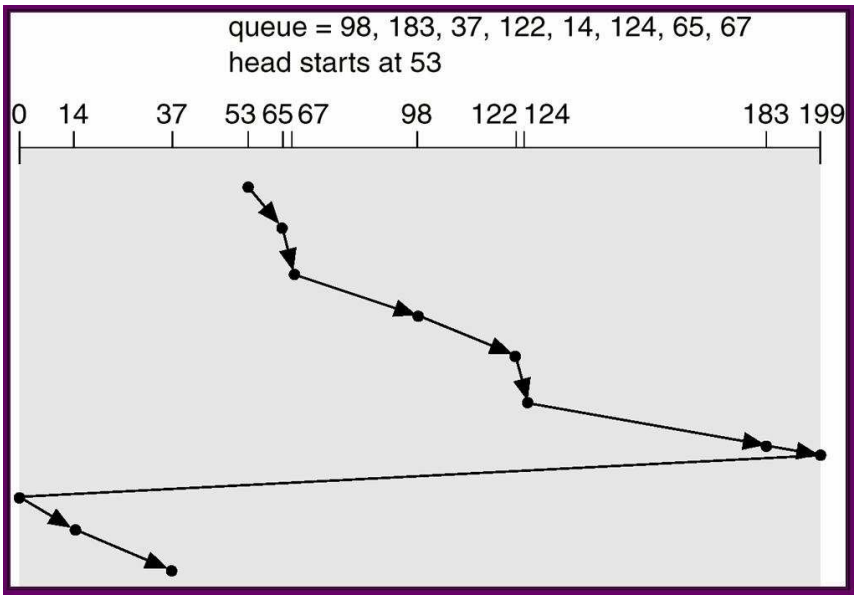
디스크 스케줄링 (계속)

Slide 8

- 순환 SCAN(C-SCAN) 스케줄링
  - 대기 시간을 보다 균등하게 하기 위한 SCAN 스케줄링의 변형
  - SCAN 스케줄링에서는 방향의 반대 방향 끝 쪽에 있는 것이 대기 시간이 길음
  - 이를 위해 한 쪽 방향으로만 처리, 끝에 다다르면 다시 복귀

디스크 스케줄링 (계속)

Slide 9



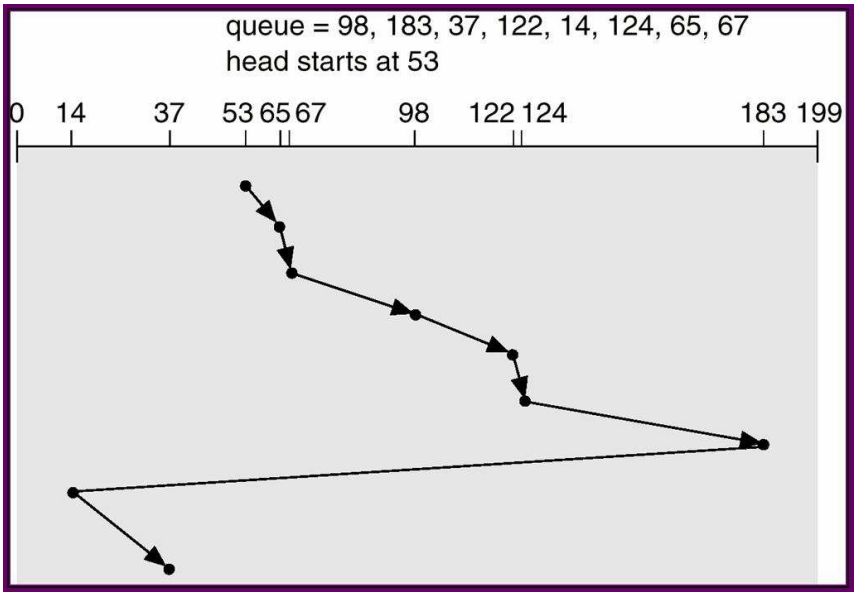
디스크 스케줄링 (계속)

Slide 10

- LOOK 스케줄링
  - SCAN 스케줄링에서는 헤드에 양쪽 끝까지 이동하나
  - LOOK 스케줄링에서는 요청의 끝까지만 이동하고 방향 전환

디스크 스케줄링 (계속)

Slide 11



디스크 관리

Slide 12

- 디스크 포맷팅
  - 새로운 자기디스크는 기록물질만 있는 텅빈 판임
  - 자료를 저장하기 전에 섹터로 나누어야함 이를 저급 포맷팅(low-level formatting)이라함
  - 섹터는 헤더 + 자료영역(보통 512B) + 오류수정코드(ECC: error correcting code)로 구성됨
  - 섹터에 쓸때 자료영역을 이용하여 ECC를 계산하여 ECC에 기록
  - 섹터에서 자료를 읽을 때 읽은 자료로 구한 ECC 와 ECC에 기록되어 있는 ECC를 비교

  ➥ 틀리면 오류있는 것임
  - 저급 포멧팅에서는 자료영역에 대한 크기를 지정가능 (보통 256B, 512B, 1KB)

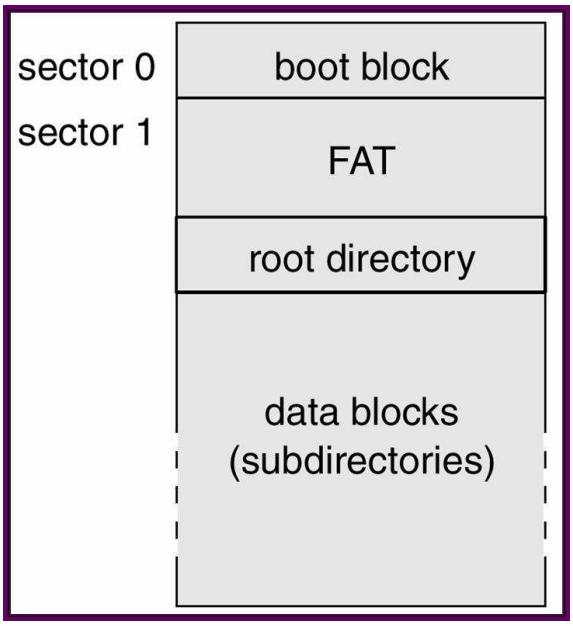
디스크 관리 (계속)

Slide 13

- 디스크 파티션
  - 하나의 디스크를 하나 이상의 집합으로 나눔
  - 각 파티션은 독립적인 디스크로 취급됨
- 논리적 포맷팅
  - 파일시스템을 만드는 과정
  - 운영체제가 초기 파일 시스템 자료를 디스크에 저장
  - 디스크의 가용공간 및 할당공간의 매핑 및 초기 빈 디렉토리 정보등이 기록
- 부트 블록
  - 부트스트랩 프로그램
    - 전원을 켜거나 재부팅시 맨 처음 실행되는 프로그램 (보통 ROM에 저장됨)
    - 중앙처리 장치 및 기타 장치제어기등을 초기화 후 운영체제 적재 및 실행
    - 부트스트랩 프로그램의 교체를 편리하게 하기 위해
      - ROM에서는 간단한 작업만하고
      - 나머지 작업은 부트 블럭에 저장
    - 부트 프로그램의 변경은 부트블록을 수정하여 가능

디스크 관리 (계속)

Slide 14



디스크 관리 (계속)

Slide 15

- 손상 블록
  - 손상된 블록은 사용되지 않게 해야함 (보통 출고시에도 있음)
  - MS-DOS에서는 format이 손상블록을 발견하면 FAT내에 이를 기록하여 사용하지 못하게함
  - SCSI용 디스크에서는 여분의 섹터를 관리하여 손상이 발견되면 대체함
  - 그러므로, 손상 블록이 많을 경우 디스크 스케줄링 효율에 문제가 될 수 있음

교체공간 관리

Slide 16

- 디스크 상에 스왑핑용 공간은 크게 두가지 방법으로 구현
  - 큰 파일을 사용하는 방법
    - 파일 시스템 관리하에 있음
    - 파일 시스템 거쳐야함으로 속도가 느려질 가능성이 있음 (디렉토리 검색 및 디스크 할당 자료구조 추적등)
    - 쉽게 크기를 조정할 수 있음
    - 윈도우에서 사용하는 방법
  - 별개의 디스크 파티션을 사용하는 방법
    - 파일시스템이나 디렉토리가 존재하지 않아서 속도가 빠름
    - 크기를 조절하기가 쉽지 않음 (파티션을 조절해야하기 때문)

RAID

Slide 17

- RAID 란?
  - 예전 약어: Redundant Arrays of Inexpensive Disks
  - 근래 약어: Redundant Arrays of Independent Disks
  - 여러개의 디스크를 사용하여 고속의 신뢰성있는 대용량 디스크를 구현
- 여분(redundant)의 디스크를 사용하여 신뢰성 개선
  - 하나의 디스크의 MTTF(mean time to failure)가 100,000 시간일 때
  - 100개의 디스크 어레이의 MTTF는 100,000 / 100 = 1,000 시간

➥ 고속의 (병렬로 동작) 대용량 디스크 구현이 가능하나 신뢰성이 떨어짐

➥ 여분의 디스크를 사용하여 신뢰성을 높임

RAID (계속)

Slide 18

- RAID level 1 (mirroring (or shadowing))
  - 가장 간단한 방법, 모든 디스크를 중복시킴
  - 하나의 논리 디스크는 두개의 물리 디스크로 구성됨
  - 모든 쓰기는 두 디스크에 상에서 수행됨
  - 데이터 손실은 첫 오류디스크의 교체전 두번째 디스크의 오류시에만
  - MTTF는 두 디스크의 MTTF 와 MTTR(mean time to repair)에 의존함
  - 각 디스크의 오류가 독립적이라면
    - MTTF가 100,000시간이고 MTTR이 10시간일 경우
    - MTTDL(mean time to data loss)는 $100{,}000^2$ / (2*10) = 500×10^6 시간 (57,000년)
  - 보통 오류는 독립적이지 않음
    (전원나감, 화재, 침수등은 두 디스크 동시 오류발생)
  - 또한, 디스크 사용시간이 길어질 수록 오류 확률증가
  - 동시에 두개의 디스크에 쓸때 오류발생 →두 디스크 모두 오류
  - 보통 하나 먼저쓰고 하나는 이어서 씀

RAID (계속)

Slide 19

- 병렬성을 통한 성능의 개선
  - 다중 디스크의 병렬 접속을 통한 이득
    - mirroring 에서
      - 읽기 요청은 두개의 디스크 각각으로 나뉠 수 있음
      - 디스크 하나당 디스크 읽기 처리는 동일하나
      - 단위 시간당 처리되는 읽기의 수는 두배가 됨
  - 전송률의 개선
    - 다중 디스크에 데이터를 스트라이핑(striping)함
    - data striping: data의 각 비트를 각 디스크에 저장 (bit-level striping)
    - 예)
      - 8개의 디스크에 각 바이트의 해당 비트를 저장
      - 섹터의 크기가 8배 커진 하나의 디스크로 처리됨
      - 모든 디스크는 모든 읽기/쓰기에 참여함
      - 초당 처리되는 접속수는 동일하나 8배 많은 데이터가 처리됨
    - striping 의 크기를 블록으로 하면 →block-level striping

RAID (계속)

Slide 20

* 예)
  · n개의 디스크에 하나의 파일이 스트라이핑될 경우
  · 블록 i는 (i mod n)+1 디스크에 저장됨
* 기타 striping 의 단위
  · 섹터의 바이트
  · 블록의 섹터등도 가능

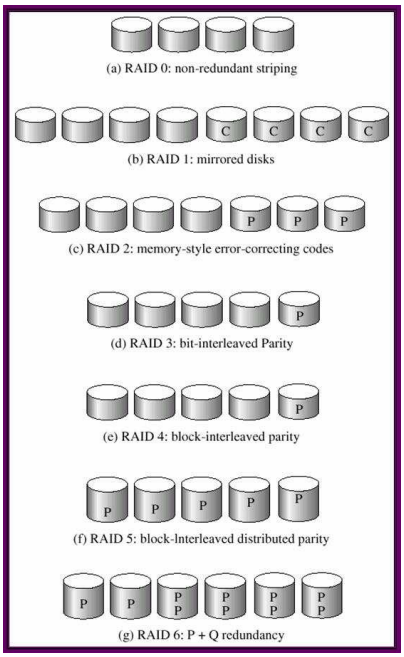
RAID (계속)

Slide 21

- RAID 레벨
  - mirroring 은 고신뢰성을 주나 비용이 비쌈
  - striping 은 고전송률을 주나 신뢰성을 높이지 못함
  - RAID 구현: disk striping, redundancy 와 더불어 parity bit를 이용
  - P는 parity bit C는 데이터 copy를 의미
  - 레벨 0: block-level striping, no redundancy
  - 레벨 1: disk mirroring
  - 레벨 2: memory-style error-correcting-code (ECC)

RAID (계속)

Slide 22



RAID (계속)

Slide 23

- 레벨 3: bit-interleaved parity
  - 하나의 디스크가 오류시 패리티 비트로 복구가능
  - 레벨 2와 동일기능이나 패리티를 위해 하나의 디스크 사용
    (레벨 2는 실제적으로 사용되지 않음)
  - 레벨 1에 비교하여 장/단점
    - 추가 디스크가 적음
    - 읽기/쓰기가 모든 디스크에 걸쳐 읽어남
      전송률이 N배 빨라짐
    - 그러나 모든 디스크가 동작함으로 처리율은 낮음
    - 또한 패리티 비트 계산에 고비용
      (패리티 계산 전용 하드웨어를 사용해서 해결)

RAID (계속)

Slide 24

- 레벨 4: block-interleaved parity
  - 레벨 0와 같이 block-level striping 사용
  - 단, 추가 디스크에 패리티 블록을 저장
  - 하나의 디스크에 오류가 발생하면 패리티 블록을 이용하여 복구
  - 블록 읽기는 하나의 디스크만 동작시킴 그러므로 전송률은 낮음
  - 단, 여러개의 읽기처리는 병렬로 처리될 수 있음 (처리율 향상)
  - 많은 양의 읽기나 쓰기는 높은 전송률을 보임
  - 적은 양의 독립적인 쓰기는 병렬적으로 수행되지 않음
    (왜냐하면 모든 쓰기에서 패리티블록이 다시 쓰여짐으로)
    (쓰여질 블록과 패리티블록은 먼저 읽혀져서 새로운 패리티블록을 계산함)

RAID (계속)

Slide 25

- 패리티 블록 계산 방법

$$X_4(i) = X_3(i) \oplus X_2(i) \oplus X_1(i) \oplus X_0(i)$$

- 여기서 $X_1(i)$ 의 내용이 $X_1^{'}(i)$로 갱신될 경우에 갱신후 패리티 비트 $X_4^{'}(i)$는

$$\begin{aligned} X_4^{'}(i) &= X_3(i) \oplus X_2(i) \oplus X_1^{'}(i) \oplus X_0(i) \\ &= X_3(i) \oplus X_2(i) \oplus X_1^{'}(i) \oplus X_0(i) \oplus X_1(i) \oplus X_1(i) \\ &= X_4(i) \oplus X_1(i) \oplus X_1^{'}(i) \end{aligned}$$

- 그러므로 갱신전 데이터와 갱신전 parity 및 갱신 데이터를 이용하여 새로운 패러티 비트 계산가능
- 결국 갱신의 경우에는 두번 읽기 와 두번 쓰기의 동작이 필요

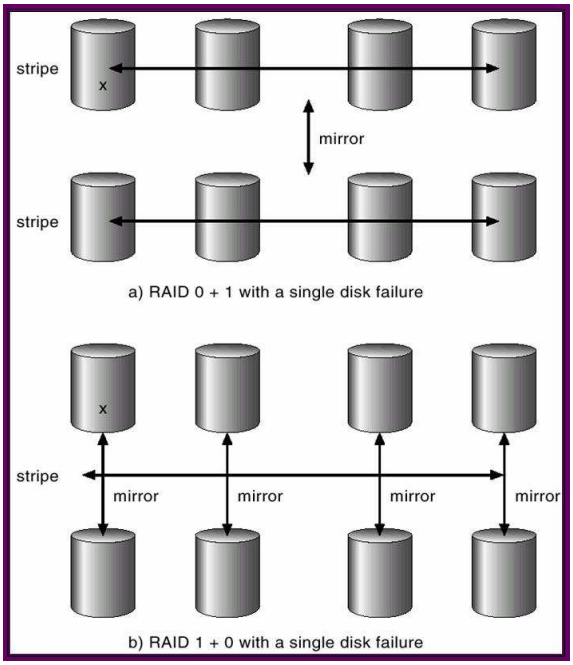
RAID (계속)

Slide 26

- 레벨 5: block-interleaved distributed parity
  - 레벨 4와는 다르게 데이터와 패리티를 $N+1$ 디스크에 분산 저장함
  - 각 블럭별로 하나의 디스크에 패리티가 나머지 디스크에 데이터가 저장됨
  - 예로, n번째 블럭은 (n mod 5) + 1 디스크에 패리티를 나머지에 데이터를 저장
  - 레벨 4가 패리티 디스크에 과부담을 주는데 비해 레벨 5는 부담이 분산됨
- 레벨 6: P+Q redundancy scheme
  - 레벨 5와 유사함
  - 1개 이상의 디스크 오류시에도 대응할 수 있도록 추가 여분 정보를 저장
  - 추가여분 정보는 패리티가 아니라 오류정정용 Reed-Solomon 코드를 사용
  - 그림에서는 모든 4비트 데이터에대해 2비트 여분데이터를 저장
    ➥두개의 디스크 오류에도 대처할 수 있음
- 레벨 0+1
  - RAID 0 (고성능) + RAID 1 (고신뢰성)
  - 비용은 RAID1과 같이 고비용
  - RAID 0+1: 스트라이핑된 디스크를 미러링함
  - RAID 1+0: 미러링된 디스크를 스트라이핑함

RAID (계속)

Slide 27



RAID (계속)

Slide 28

- hot space
  - 디스크 오류를 빠르게 교체하기위한 교체용 예비 디스크
  - RAID에서 하나의 디스크가 오류시 바로 자동으로 예비 디스크로 교체

디스크 부착

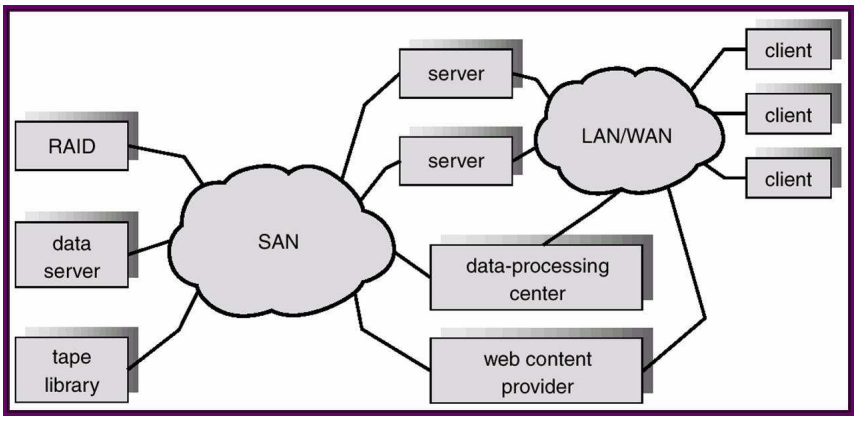
- 컴퓨터에 디스크를 부착하는 두가지 방법
  - I/O 포트를 사용하는 방법 (호스트-부착 저장장치)
  - 분산 파일 시스템을 사용하는 방법 (네트워크 부착 저장장치)

Slide 29

- 호스트-부착 저장장치 (host-attached storage)
  - 지역 I/O 포트를 통해 부착
  - I/O 버스를 사용한 방법
    - IDE 나 ATA: 최대 I/O 버스당 2개 드라이브 연결가능
    - SCSI 와 FC(fibre channel)
  - FC switch 를 이용한 부착 →SAN(storage-area networks)

디스크 부착 (계속)

Slide 30



디스크 부착 (계속)

- 네트워크-부착 저장장치 (NAS: network-attached storage)
  - 네트워크를 통해 저장장치 연결
  - NAS를 하나의 저장장치-접속 프로토콜로 간주할 수 있음
    - 호스트-부착 저장장치: SCSI 장치 구동기 및 SCSI 프로토콜 사용
    - 네트워크-부착 저장장치: TCP/IP 상에 RPC 사용

Slide 31