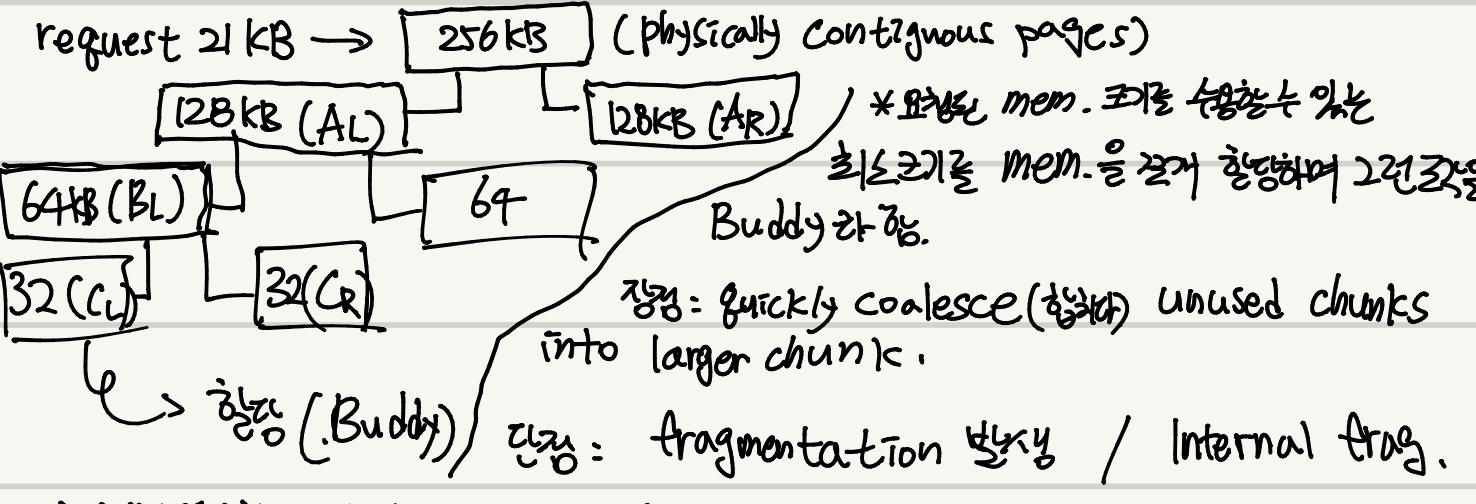


② Memory-mapped files: allows file I/O to be treated as routine mem. access by mapping a disk block to a page in mem.
→ open, read, write system calls can map file pages to memory ⇒ 파일의 디렉토리 access와 sys. call 결과 ⇒ 가능하다. sys. proc의 가능성을 파일의 파일의 sys. call 결과 mem. 을 접근. ⇒ Also allow several procs to map the same file allowing the pages in mem. to be shared.

When does written data make it to disk? ⇒ sys. or file 'close()' time.
② Allocating kernel Mem.: Treated Differently from user mem.

Often allocated from a free-mem. pool. (Kernel requests mem. for structures of varying sizes - minimizing waste due to fragmentation / Some kernel mem. needs to be contiguous. I.e. for device I/O that directly interact with physical mem.)

③ Buddy Sys.: Allocates mem. from fixed size segment consisting of physically-contiguous pages. (Power of 2 for kernel proc. on allocating)



→ 디스크에 대한 접근도 가능. 400 kB 요청에 대해 128 kB, 256 kB 블록을 반복하여 사용
이유는 병합할 수 있는 블록으로 확장할 수 있는 가능성이 높아짐.

④ Slab Allocator: alternate strategy; slab은 physically contiguous pages로 허용

Cache consists of one or more slabs; 각각은 커널 구조에 대해 Single Cache 구조 ⇒ 흥미는 type(size)의 모든 객체를 하나의 구조체에 할당하는 드리; 각 개체는 object (instance of data structure)로 처리됨. ; 사용자 프로그램은 object를 malloc; struct, & store 할 때 object는 used로 표시되며; slab은 used 아니면 가능하면 다른 가능한 empty slab으로 쓰임. 만약 empty slab 없으면 new slab 생성; 증명은 "No Fragmentations, fast mem. request satisfaction."

⑤ Prepaging: proc. 시작할 때 모든 디렉토리 PF 초기화. 필요한 것들은 모든 page를 ref. 되기 전에 미리 로드해. 문제로드가 일어나면 I/O, mem. 낭비.

증명: 50% page가 prepage. 대기 사용 ratio. Prepaging 된 page는 PF 접근 비용 5x로↓ 불필요하게 디렉토리에 접근 비용 5(1→5)보다 나중에 바로↓ 가능해짐.

⑥ Page Size: ① Frag. 증가 ② Page table size 증가 (→ 필요한 페이지 개수와 테이블 사이즈 증가↑)

③ Resolution 감소 (Coarse resol.) ④ I/O overhead 증가 (→ 초기화)

처리하는 데이터 빙어지거나) ⑤ PF 감소 ⑥ Spatial locality 증가 (→ 처리하는 mem. 공간에 대해서는 디렉토리에 접근 비용↓) ⑦ Temporal locality 증가 (→ 처리하는 데이터 종류가 같은 경우에 처리되는 차례를 기반. 성능향상)

⑧ TLB entry가 처리하는 mem. 주소→주소 ⇒ TLB miss 가능성.

⑨ TLB reach: The amount of mem. accessible from TLB = (TLB size * Page size). ① 아동적 상황: 각 proc.의 Working Set이 TLB에 저장되어있어야.

② Page size 증가시 TLB reach 증가. (⇒ Frag. 증가 가능성). ③ 다양한 page size 제공. ⇒ 한 page size를 필요로 하는 application이 Frag. 가능성의 사용 가능. 이를 위해 OS가 TLB 주소에 따라, one of fields in a TLB entry must indicate the size of page frame.

⑩ program structure: Each row is stored in one page if page size is 128kB.

* int [128, 128] data; for(j=0; j<128; j++) {for(i=0; i<128; i++) {data[i][j]=0;}}

⇒ 128x128 faults.

for (i=0; i<128; i++) {for(j=0; j<128; j++) {data[i][j]=0;}} ⇒ 128 PF.

⑪ I/O interlock: page는 디렉토리 mem.의 lock(lock) 상태에.

① I/O 중지됨) page 고려되었을 때, 디렉토리 전송 충돌 가능성. ② 각 frame에는 lock bit →交错로 처리. frame의 lock 상태에 따라 처리되는 차례로. ③ DB proc. 같은 User proc.는 lock pages into mem. 필요로. ④ 동기화 pending.

4. File System.

① 파일연관동안 File System이 파일연관 정보를 주요로.

② Open file table (열려있는 파일을 관리) ③ File pointer (proc.의 파일 열려있는 상태 proc.의 대해서 미리작성으로 원자성을 가진다는 pointer)

④ File Open Count (열린 횟수 count) ⑤ Disk location of file (Data access information + cache) ⑥ Access rights.

⑦ File: is a named collection of related information that is recorded on secondary storage.

⑧ Access Method ① Sequential Access: 파일 데이터를 순서대로 r/w.

read next; write next; reset; ② Direct Access: file is fixed length logical records // read n; write n; position to n; read next; write next; ; n is relative block number

⑨ Directory: collection of nodes containing information about all files. * Both Dir. structure and files reside on disk.

⑩ Disk Structure: Disk partition or logical partition; Disk or Partition은 RAID에 포함 (Failure로 인해 파일이 더 이상 접근할 수 없음 ⇒ 디스크 손상 방지) ; Disk or Partition can be used raw - without a file system, or formatted with a file system; Entity containing file sys. known as volume; Device directory or Volume table of contentor file sys.의 정보를 기록한다.

⑪ Directory Organization: Dir.은 내용을 관리하는 높은 단위로 구성된다.

① Efficiency (파일 복제하기) ② Naming (User 관리하기): 같은 이름 file을 다른 dir.에 저장하거나, 하나 file이 여러 이름 지정 가능) ③ Grouping: 그룹화 그룹화하기.

④ Single Level Directory: Single Dir. for users. ⇒ ① 모든 file 이름이 대체로 가능. Naming problem. ② Logically grouping이 어렵다: Grouping problem.

⑤ Two Level Directory: 각 User 별로 Dir.를 별개로. User별 파일 분포는 대체로 2개의 Sub Dir.로. No grouping capability / Efficient searching.

⑥ Tree Structured Dir: Efficient Searching; Grouping Capability; Current Dir.의 정보를; Do not allow sharing file/Dir; Absolute or relative file name 처리 - Dir.의 처리 > child subtree 처리.

⑦ Acyclic-Graph Dir: Subdir.의 file 접근 가능; Deleting: Two diff. names are possible for a single file or dir; Bug: Dangling pointer (A pointer that directs a removed file or dir.) Sol: ① By keeping backpointers, OS can delete all pointers that direct a file or directory that is to be deleted.

⑧ By keeping a counter of pointers in a file or a dir., the deleting operation decreases the counter and the file is deleted if the counter becomes zero.; New dir. entry type & link: another name(pointer) to an existing file ② Resolve the link: follow pointer to locate the file.

⑨ General Graph Dir: 사이클 있음. 종종으로 두거나 만들기 가능: ambiguous access to files and/or dirs. ; No cycle 보장하지만? Sub Dir.의 obj file은 link를 허용. 이를 link로부터 cycle detection alg. 사용; Garbage Collection: 1st. traverses all files and dirs. with marking them, and the 2nd, scans files and dirs. without markings to reallocate them.

⑩ File System mounting: A file system must be mounted before it can be accessed.

; An unmounted sys. is mounted at a mount point.

⑪ File Sharing: On distribution sys. file may be shared across a network; Network File System (NFS) is a common distributed file-sharing method. ; Uses networking to allow file sys. access between systems. Manually via programs like FTP. Automatically) Seamless (운영체계) Using distributed file sys. ; Semi automatic via world wide web.

⑫ protection: chmod (761) game. rwx (421)
↳ (= rwx r-- --x)

⑬ File System Structure: file sys. to Secondary Storage (disk) or residing in:

→ ① provides an user interface to storage, mapping (logical to physical) ② provides efficient and convenient access to disk by allowing data to be stored, located, and retrieved easily. ; File Control Block (FCB): Storage structure consisting of information about a file. ; Device driver: Controls physical device

⑭ Layered File sys.: 각 층은 다른 층의 기능을 사용. 서로 다른 인터페이스를 제공 (app. programs) → (local file sys.) → (file organization module) → (basic file sys.)

→ (I/O control) → (devices) ; Device driver manages I/O devices at I/O control layer; Basic file sys. translates given commands like "retrieve block 128" to device driver; File org. module understands files, logical addr., and physical blocks; Logical file sys. manages metadata info.

⑮ Virtual File System (VFS): acts as an intermediate layer that connects general file operation to heterogeneous file sys. and environments.

⑯ Allocation Method: ① Contiguous Allocation: Each file occupies set of contiguous blocks. ; the case of sequential access (sequential access) with regard to contiguity. Simple - Only starting location (block#) and length (number of blocks) are required. ; problems include finding space for file, knowing file size, external frag., need for compaction off-line (down time) or on-line.

* 내부적으로 발생하는 미세한 불均衡이 파일 크기마다 큼.

② Extent-based System: allocate disk blocks in extents (is a contiguous block of disks) ⇒ 4/외부 단편화 가능.

1) Internal frag.: 파일이 70kb, 80kb로 같은 extent가 128kb면, 58kb로 2) External: 파일이 70kb/80kb로 같은 extent는 extent들간에 광범위한 단편화가 있다. 두개의 70kb extent가 있을 때 70kb로 80kb로 200kb로 Extent로 단편화가 있고, else External frag. ob.

③ Linked Allocation: Each file keeps a linked list of blocks; File ends

at nil pointer (null instance pointer); No External frag. ;

단점: ① Direct access 가능. ② 파일연관성이 필요 ③ 높은 디스크: 충분히 큰 디스크에 접근. 이를 단점이 될 가능. ④ 불균형이 충분히 허용될 경우, 속도 느려짐.

* FAT (File Allocation Table): file system, file와 dir.를 디렉토리 구조에 청탁하는 계약이며, 이를 dir. entry라 부른다. 이는 sys.에서 FAT는 Disk의 사용부분이 위치한다. (Linked list, 연결리스트로 파일연관은 FAT를 활용함 ⇒ Direct access 가능.)

④ Indexed Allocation: Each file has its own index block(s) of pointers to its data blocks. ;

1) Logical View: file system의 directory entry는 file의 index block 위치이다; Index block은 file의 data block의 주소를 포함한다.

2) Random access: indexed allocation은 모든 block의 index block에 대해 고려되는 Direct access 가능.

3) Index block이 파일을 순서대로 처리하기 가능하고, 연속으로 파일을 찾으려면 External frag. 가능. , Index table 필요.

4) 단점은 작은 초기 파일에서도 하나의 블록을 인접으로 쓰다가 차츰 공간이 채워지면, 하나의 index block은 큰 파일을 처리할 수 없음 (Ex. 1 block 512kb면 최대 사용되는 block의 index 개수가 512/4byte(파일크기)= 128개다. 즉 파일크기 2GB= 128*512byte(block 크기)= 64KB. 1KB의 1KB여도 1000/4= 256개 ⇒ 256KB다).

Sol. ① Linked Scheme: Index block이 2진법으로 만들어 linked allocation 한다. 즉,

각 index block은 다음 index block에 대한 pointer를 가진다.

② Multi-level Scheme: 파일은 높은 단위로 처리, Index block은 모든 파일에 대해 처리된다. Ex. 최대의 Index block이 2^10개로 만든다. 즉 1024개의 Index block이 2^10개로 만든다.

즉 파일은 중간 단위로 Index block이 가진다. 중간 단위로 Index blocks는 2^10개로 만든다. 즉 1024개의 Index block은 2^10개로 만든다.

즉 2^10*2^10 = 2^20개 data block로 처리된다.

⑯ Free Space Management

① Bit map or Bit vector: 각 block은 1bit로 표시. (비트맵)

: 부사 공간 필요, 연속적인 n개 free block 찾기 어려움 (Ex. block size= 4KB= 2^12 bytes, disk size = 2^30 bytes (1TB) ⇒ n = 2^30/2^12 = 2^18 bytes (32MB), If we use clusters of 4 blocks → 8MB of mem.

② Linked List: contiguous space를 찾기 쉽지 않음, 공간 낭비가 많다. No need to traverse the entire list (if # free blocks recorded)

③ Grouping: linked list 형태. n개의 free-block을 grouping하고 이를 linked list로.

contiguous free space 찾기 쉬운 점.

④ Counting: 연속성, extent, clustering된 disk 공간을 관리하는 방법이다.

Basic idea: 1st 번째 주소와 2위 연속된 번째 주소 차이. (연속된 주소들와 그 이후 연속된 번째 주소 간의 차이)

INDEX.

(2019) ① Inverted PT+ smallest PT list.

② Hash table hash table page number.

③ Slab allocation alg: partial slab → empty slab → create a new slab and return one object of the slab.

④ physical address = relocation reg. + logical add. (execution time binding - tlb)