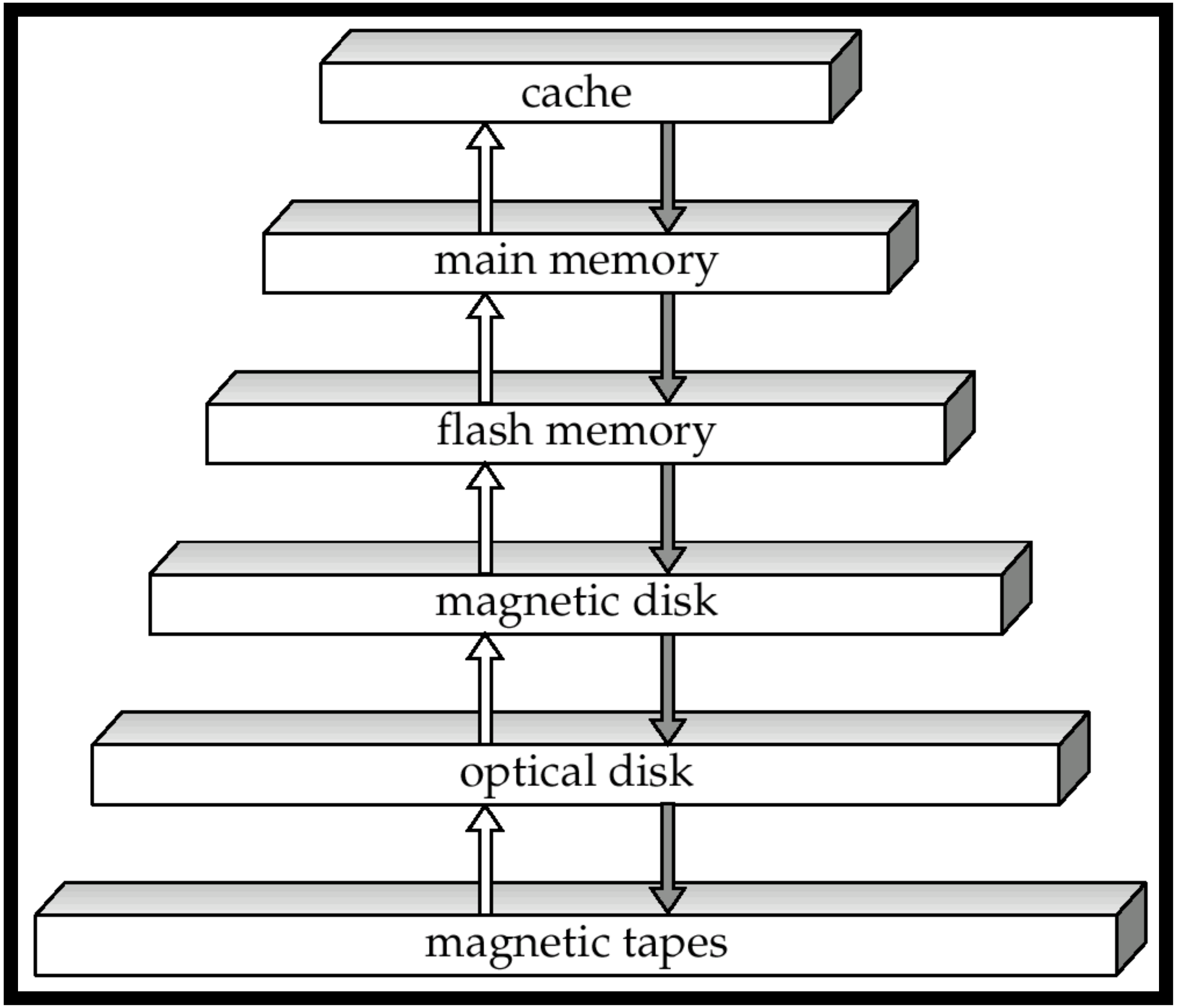
CMSC 424 – Database design
Lecture 12
Storage

Mihai Pop

Administrative

- Office hours tomorrow @ 10
- Midterms are in – solutions for part C will be posted later this week
- Project partners – I have an odd number of people...

Storage Hierarchy



Storage Hierarchy

- Cache - Super fast; volatile
- Main memory - 10s or 100s of ns; volatile
- Flash memory - limited number of write/erase cycles; non-volatile, slower than main memory
  - Intel announcement
- Magnetic Disk - Non-volatile
- Optical Storage - CDs/DVDs; Jukeboxes
- Tape storage - Backups; super-cheap; painful to access

1956
IBM RAMAC
24” platters
100,000 characters each
5 million characters

1979
SEAGATE
5MB

1998
SEAGATE
47GB

2004
Hitachi
400GB
Height (mm): 25.4. Width (mm): 101.6. Depth (mm): 146. Weight (max. g): 700

2006
Western Digital
500GB
Weight (max. g): 600g

2007

http://www.engadget.com/2007/01/05/hitachi-breaks-1tb-hard-drive-barrier-with-7k1000/

le Ca... | Remembe... | Graph Clu... | Gaggle | PLoS Com... | Reviewer:... | ARDB-Anti... | Google News | Syllabus f...

WoW players: we have all your patch 2.4 news!

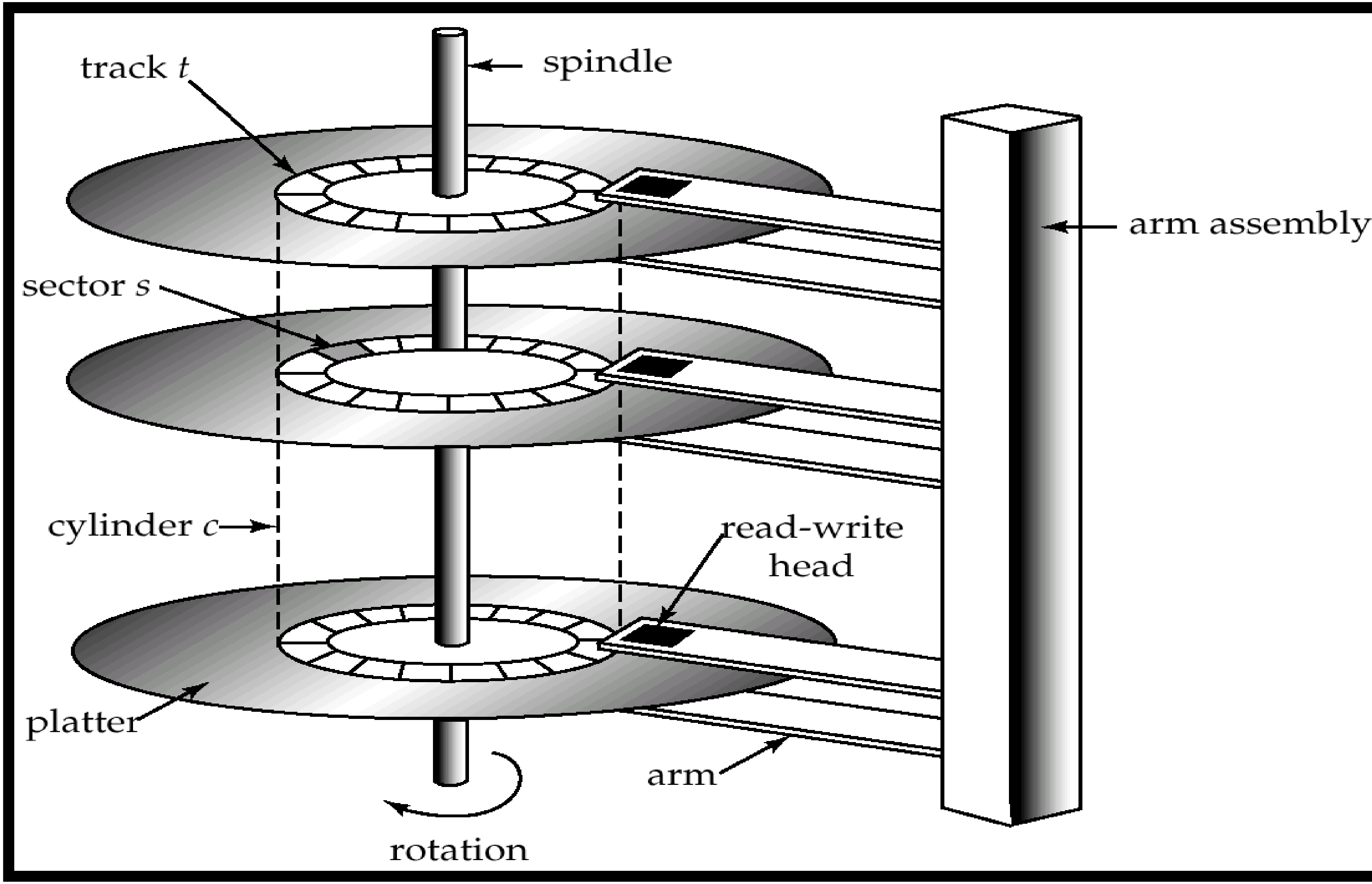
engadget

Hitachi breaks 1TB hard drive barrier with 7K1000

Posted Jan 5th 2007 2:12AM by Ryan Block
Filed under: CES, Storage

Well, we knew it was likely to happen in 2006 or 2007: Hitachi has fulfilled their promise and broken the 1TB drive barrier with the introduction of Hitachi's new Deskstar 7K1000 drive. Thanks to perpendicular recording and the average consumers' voracious appetite for ~~porn~~ totally legitimate data, Hitachi's new $400 drives -- available in SATA II or PATA 133 varieties, with differing speed modes, a 32MB buffer, quieting accoustics, SMART, and a 7200rpm spindle speed -- will hit the market running in Q1 of this year. Also announced: the CinemaStar 7K1000, a DVR-centric drive due in Q2 which wasn't fully detailed, but apparently has "adaptive error recovery", "Smooth Stream Technology to optimize the drive for audio/video applications requiring reliable storage", and other buzzy sounding stuff which just seems a lot like regular old drive features. We'll assume it's better tuned for high-throughput read / write performance, and leave it at that.

Tags: 1tb, 7K1000, breaking news, BreakingNews, cinemastar, deskstar, hitachi



track t
spindle
arm assembly
sector s
cylinder c
read-write
head
platter
arm
rotation

Performance Measures of Disks

- **Access time** – the time it takes from when a read or write request is issued to when data transfer begins. Consists of:
  - **Seek time** – time it takes to reposition the arm over the correct track.
    - Average seek time is 1/2 the worst case seek time.
      - Would be 1/3 if all tracks had the same number of sectors, and we ignore the time to start and stop arm movement
    - 4 to 10 milliseconds on typical disks
  - **Rotational latency** – time it takes for the sector to be accessed to appear under the head.
    - Average latency is 1/2 of the worst case latency.
    - 4 to 11 milliseconds on typical disks (5400 to 15000 r.p.m.)
- **Data-transfer rate** – the rate at which data can be retrieved from or stored to the disk.
  - 25 to 100 MB per second max rate, lower for inner tracks
  - Multiple disks may share a controller, so rate that controller can handle is also important
    - E.g. ATA-5: 66 MB/sec, SATA: 150 MB/sec, Ultra 320 SCSI: 320 MB/s
    - Fiber Channel (FC2Gb): 256 MB/s

Reliability Issues:

Mean time to failure (MTTF):

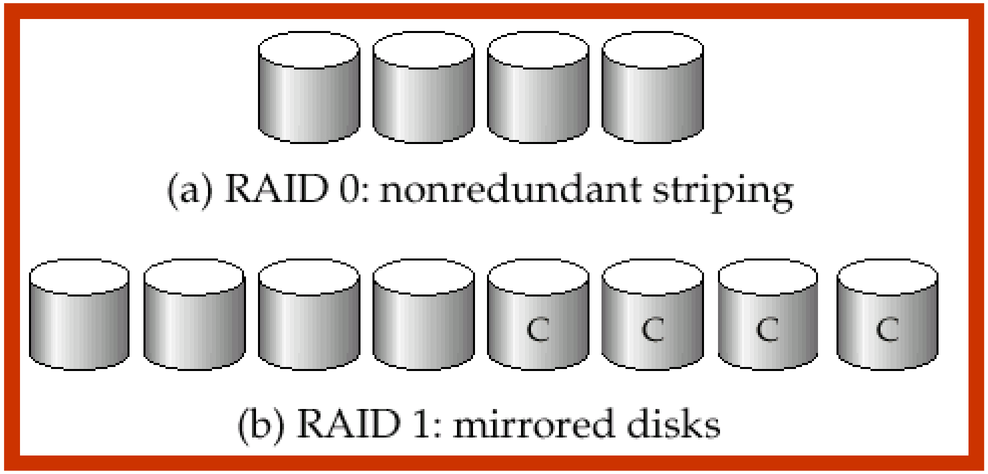
57 to 136 years

Given 1000 new disks with 1,200,000 hours of MTTF, on average one of them will fail in 1200 hours = 50 days.

RAID Levels

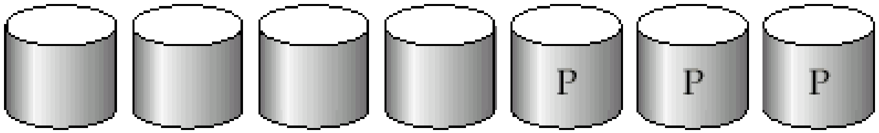
- Schemes to provide redundancy at lower cost by using disk striping combined with parity bits
  - Different RAID organizations, or RAID levels, have differing cost, performance and reliability characteristics

- **RAID Level 0**: Block striping; non-redundant.
  - Used in high-performance applications where data lose is not critical.
- **RAID Level 1**: Mirrored disks with block striping
  - Offers best write performance.
  - Popular for applications such as storing log files in a database system.

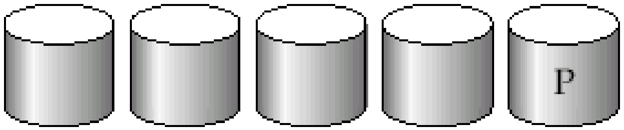


RAID Levels (Cont.)

- **RAID Level 2**: Memory-Style Error-Correcting-Codes (ECC) with bit striping.
- **RAID Level 3**: Bit-Interleaved Parity
  - a single parity bit is enough for error correction, not just detection, since we know which disk has failed
    - When writing data, corresponding parity bits must also be computed and written to a parity bit disk
    - To recover data in a damaged disk, compute XOR of bits from other disks (including parity bit disk)



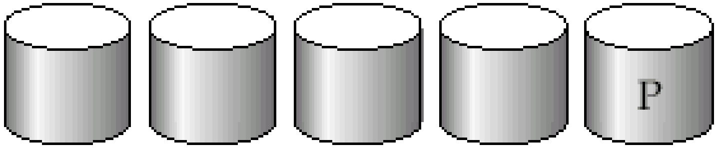
(c) RAID 2: memory-style error-correcting codes



(d) RAID 3: bit-interleaved parity

RAID Levels (Cont.)

- RAID Level 3 (Cont.)
  - Faster data transfer than with a single disk, but fewer I/Os per second since every disk has to participate in every I/O.
  - Subsumes Level 2 (provides all its benefits, at lower cost).
- **RAID Level 4:** Block-Interleaved Parity; uses block-level striping, and keeps a parity block on a separate disk for corresponding blocks from *N* other disks.
  - When writing data block, corresponding block of parity bits must also be computed and written to parity disk
  - To find value of a damaged block, compute XOR of bits from corresponding blocks (including parity block) from other disks.



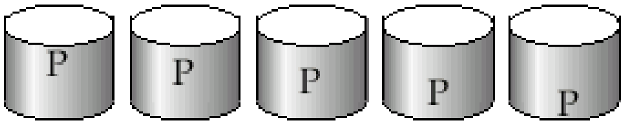
(e) RAID 4: block-interleaved parity

RAID Levels (Cont.)

- RAID Level 4 (Cont.)
  - Provides higher I/O rates for independent block reads than Level 3
    - block read goes to a single disk, so blocks stored on different disks can be read in parallel
  - Provides high transfer rates for reads of multiple blocks than no-striping
  - Before writing a block, parity data must be computed
    - Can be done by using old parity block, old value of current block and new value of current block (2 block reads + 2 block writes)
    - Or by recomputing the parity value using the new values of blocks corresponding to the parity block
      - More efficient for writing large amounts of data sequentially
  - Parity block becomes a bottleneck for independent block writes since every block write also writes to parity disk

RAID Levels (Cont.)

- **RAID Level 5:** Block-Interleaved Distributed Parity; partitions data and parity among all *N* + 1 disks, rather than storing data in *N* disks and parity in 1 disk.
  - E.g., with 5 disks, parity block for *n*th set of blocks is stored on disk (*n mod* 5) + 1, with the data blocks stored on the other 4 disks.

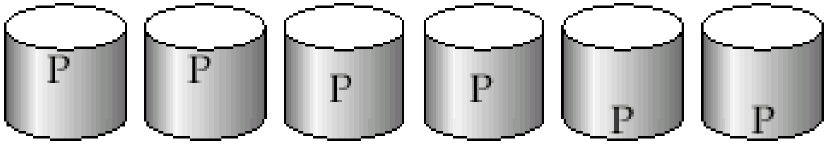


(f) RAID 5: block-interleaved distributed parity

P0	0	1	2	3
4	P1	5	6	7
8	9	P2	10	11
12	13	14	P3	15
16	17	18	19	P4

RAID Levels (Cont.)

- **RAID Level 5 (Cont.)**
  - Higher I/O rates than Level 4.
    - Block writes occur in parallel if the blocks and their parity blocks are on different disks.
  - Subsumes Level 4: provides same benefits, but avoids bottleneck of parity disk.
- **RAID Level 6**: P+Q Redundancy scheme; similar to Level 5, but stores extra redundant information to guard against multiple disk failures.
  - Better reliability than Level 5 at a higher cost; not used as widely.



(g) RAID 6: P + Q redundancy

Optimization of Disk-Block Access

- Block – a contiguous sequence of sectors from a single track
  - data is transferred between disk and main memory in blocks
  - sizes range from 512 bytes to several kilobytes
    - Smaller blocks: more transfers from disk
    - Larger blocks: more space wasted due to partially filled blocks
    - Typical block sizes today range from 4 to 16 kilobytes

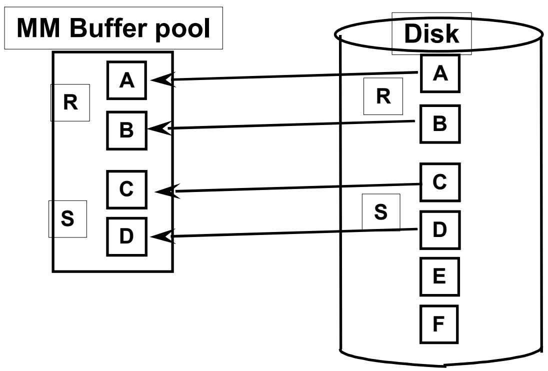
- Disk-arm-scheduling algorithms order pending accesses to tracks so that disk arm movement is minimized
  - elevator algorithm : move disk arm in one direction (from outer to inner tracks or vice versa), processing next request in that direction, till no more requests in that direction, then reverse direction and repeat
  - sequential access is 1-2 orders of magnitude faster
  - random access 10ms/1KB or 10 sec/MB as opposed to 8-10 MB/sec
  - so it pays if we combine access (elevator algorithms- piggy banking)
- <u>log-based file system</u>: does not update in-place but logs the writes to a sequential disk (achieving the sequential speeds)
- <u>clustering</u> of data: organize it to correspond to the access
  - if hierarchical access, then put the daughters next to the mothers
  - for joining tables, put the joining tuples from the two tables next to each other

Buffer Management

- the buffer pool is the part of the main memory allocated for temporarily storing disk blocks read from disk and made available to the CPU- its purpose is identical to caching for reducing I/O
- the buffer manager: the subsystem responsible for the allocation and the management of the buffer space- transparent to the users
- on a process (user) request for a block (page) the buffer mgr takes the following steps:
  - checks if the page is in the buffer pool
  - if it is, it passes its address to the process
  - if it is not, it brings it from the disk and then passes its address to the process
- very similar to the *virtual memory managers,* although it can do a lot better

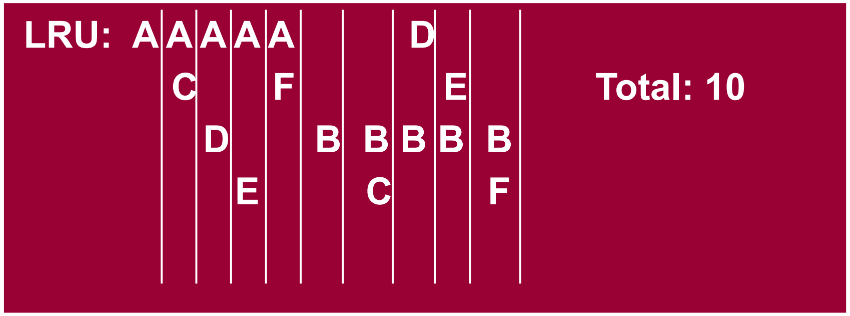
Buffer Replacement Strategies

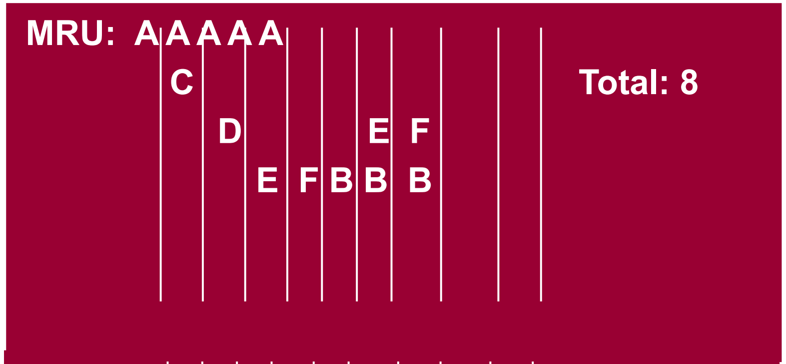
```
for each  block b(r) in  r  do begin
   for each block  b(s)  in s  do begin
       join(b(r),b(s))
   end
 end
```

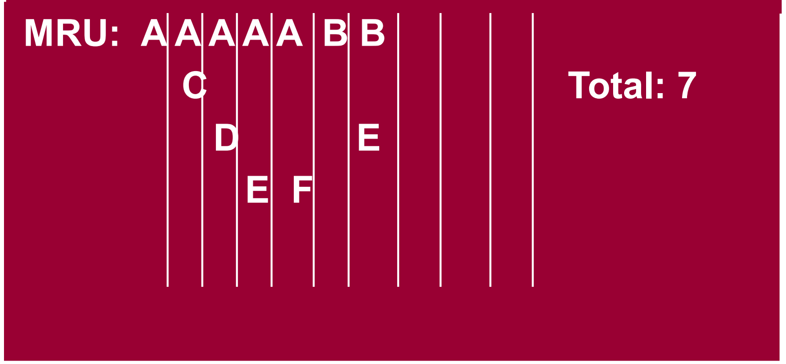


R: outer
S: inner

- LRU, FIFO, etc. used in OS do not perform well in DBMSs- MRU is better for some operations







- Some times we fasten (or pin) some blocks to keep them during the operation
- Prevents the replacement strategy to touch them until released
- These are called *fastened or pinned* blocks

Buffer-Replacement Policies (Cont.)

- Pinned block – memory block that is not allowed to be written back to disk.
- Toss-immediate strategy – frees the space occupied by a block as soon as the final tuple of that block has been processed
- Most recently used (MRU) strategy – system must pin the block currently being processed. After the final tuple of that block has been processed, the block is unpinned, and it becomes the most recently used block.
- Buffer manager can use statistical information regarding the probability that a request will reference a particular relation
  - E.g., the data dictionary is frequently accessed. Heuristic: keep data-dictionary blocks in main memory buffer

Buffer Management (cont)

- Forced output blocks: occasionally, for recovery reasons, the DBMS forces some blocks out to disk immediately (does not wait for the OS I/O scheduler)
- OS affects DBMSs operations by:
  - read ahead, write behind
  - wrong replacement strategies
  - Unix is not good for DBMS to run on top. Most commercial systems implement their own I/O on a raw disk partition
- Variations of buffer allocation
  - common buffer pool for all relations
  - separate -"- -"- each relation
  - as above but with relations borrowing from each other
  - adaptive allocation based on their needs
  - prioritized buffers for frequently accessed blocks, e.g. data dictionary
- for each buffer the manager keeps the following
  - which disk and which block it is
  - whether it was modified or not (dirty)
  - information for the replacement strategy (e.g. the time it was last accessed)