1 Blocks, Pages and Access Times

Consider a disk with a sector size of 512 bytes, 2000 cylinders, 50 sectors per cylinder, 5 double-sided platters, and a block size of 1024 bytes. Suppose that a file containing 100,000 records of 100 bytes each is to be stored on such a disk such that no record is allowed to span more than one block.

1. How many records fit onto a block?
2. How many blocks are required to store the entire file? If the file is arranged sequentially on disk (filling up one surface after the other), how many surfaces are needed? What is the optimum way to arrange the data on disk?
3. How many records of 100 bytes each can be stored using this disk?
4. Given that the time to move the head between 2 tracks is at least 4 msec whereas the head switch time is 1 msec, the average seek time is 10 msec, the disk platters rotate at 5400 rpm, and 1 track of data can be transferred per rotation: how much time is required to read the entire file sequentially?. How much time is required to read the entire file assuming random scan access? (assume both sequential and optimal arrangement of data)

Solution

1. number of records per block $= \frac{1024}{100} \approx 10$
2. number of blocks to store entire file $= \frac{100{,}000}{10} = 10,000$.
   Since number of sectors per block $= \frac{1024}{512} = 2$, we need $2 * 10,000 = 20,000$ sectors to store the entire file.
   Each track consists of 50 sectors, therefore number of tracks needed $= \frac{20{,}000}{50} = 400$.
   Thus, the number of surfaces needed is 1 (since each surface holds up to 2000 tracks).
   The optimum way to arrange the data would be to fill all tracks on one cylinder (both sides) first before moving to the next one. This would reduce the overall time spent moving the head between cylinders.
3. Since number of surfaces $= 10$, therefore the total number of tracks $= 10 * 2000 = 20,000$.
   The total number of sectors $= 20,000 * 50 = 1,000,000$.
   Thus, the total number of blocks $= \frac{1{,}000{,}000}{2} = 500,000$ block.
   The total number of records that can be stored $= 500,000 * 10 = 5,000,000$.
4. $t_{seq} = avg(t_{seek}) + t_r/2 + num\_of\_cylinders\_crossed * t_{seek} + num\_of\_head\_switches * t_{switch} + num\_of\_blocks * t_{tr}$

   $t_r = \frac{1*60*1000}{5400} \approx 11.11$ msec.

   Since each track consists of 50 sectors, and each block spans 2 sectors, therefore each track can hold up to 25 blocks.

   Since transfer rate is 1 track per rotation, and each rotation takes 11.11 msec, this implies that 25 blocks take 11.11 msec to be transfered. Therefore $t_{tr} = \frac{11.11}{25} = 0.4444$ msec.

   If we assume sequential arrangment of data on disk, the number of cylinders crossed $= 400$

and the $num\_of\_head\_switches = 0$ since we need only 1 surface to store the entire file . Thus,

$t_{seq} = 10 + 5.56 + 400 * 4 + 10,000 * 0.4444 \approx 6.06$ sec.

If we assume optimal arrangment of data on disk, the number of cylinders crossed = 40, however the number of times the head has to be switched is 399 times . Thus,

$t_{seq} = 10 + 5.56 + 40 * 4 + 399 * 1 + 10,000 * 0.4444 \approx 5.02$ sec.

$t_{rand} = num\_of\_blocks * (avg(t_{seek}) + t_r/2 + t_{tr})$
$t_{rand} = 10,000 * (10 + 5.56 + 0.4444) = 160$ sec.

2 Levels of Redundancy

Given the following configuration options for your RAID solution:

1. RAID 0
2. RAID 1
3. RAID 5
4. RAID 6

Discuss which of these RAID configurations are better in terms of: (a) read vs. write performance and (b) reliability.

Solution

1. RAID 0 - Just multiple disks. Good read/write performance due to striping, however no reliability.
2. RAID 1 - Mirrored disks (two identical copies of the data are kept in two different diks), no striping. Every write operation needs to be done in both disks. Reads can be distributed between the two disks, allowing *parallel reads*. One disk may fail without data loss. Nice for log file as there is no striping (sequential writes).
3. RAID 5 - Parity striping. Reads are faster due to parallel I/O, writes suffer because they must be decomposed into several reads and writes to keep parity information (a controller cache may hide this effect). One disk may fail without data loss (data is reconstructed from parity).
4. RAID 6 - Several independent parity codes. Read performance similar of RAID 5. Writes a bit slower than RAID 5, since the parity information now requires a larger number of disks. Multiple disks may fail without date loss (data is reconstructed from parity).

3 Sequential Scans and Extents

Define what an extent is and discuss how the distribution and sizing of extents impacts sequential scan performance. What are potential interactions with prefetching and block usage factor?

Solution

An extent is a contiguous unit of allocation for blocks in disk. The DBMS stores its objects (tables, indexes) by allocating extents, which may be of varying number of blocks. Note that in a "sequential" scan there will be a controlled amount of random I/O, as two different extents may not necessarily be sequentially laid out in disk. If extents are too small and distributed in the disk, sequential scan performance will suffer. Prefetching is a technique used to enhance sequential scan performance by bringing blocks to memory before they are requested. The idea is to perform I/O in larger chunks, thus amortizing total I/O cost. This is possible because the access pattern in a sequential scan is very well defined. If extents are as a rule smaller than the prefetch size, then the gains obtained by prefetching will be significantly smaller, as in reality the I/O for the prefetched blocks will not be sequential (but rather in smaller sized chunks). Block usage factor influences the total number of blocks allocated for a table. Therefore, large values will impact negatively sequential scan performance. Larger extents may be used to try to compensate for low usage factors.

4 Buffer Management

1. Consider the following page reference sequence:

   $$1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6$$

   How many page misses would occur using LRU, FIFO, or an optimal replacement strategy, assuming a main memory with 1, 2, 4, or 7 frames.

2. Assume variable-length pages, explain why the discussed replacement strategies might not work. Develop 2 replacement strategies based on LRU and FIFO that would work for the case of variable-length pages.

Solution

1. The number of misses are presented in the table.

Number of Frames	LRU	FIFO	Optimal
1	20	20	20
2	18	18	15
4	10	14	9
7	7	7	7

2. Since pages do not have the same length, the space occupied by the page that is chosen to be replaced may not be big enough to fit the newly allocated page. We can modify some of the strategies to take this into consideration.

LRU: Select the page that has not been used for the longest period of time and that is large enough. If no one page is large enough, select a combination of the the least recently accessed pages that are contiguous in memory (if relocation is not available) and that are large enough. If relocation is available, rearrange the least recently used N pages to be contiguous in memory and replace those with the new page.

FIFO: Find the oldest page that is occupying enough space to accommodate the new page. If relocation is not possible and no one page is large enough, select a combination of pages that are contiguous in memory, such that they are the oldest in memory. If relocation is possible, rearrange the memory so that the first N oldest pages whose space is large enough for the incoming page are contiguous in memory and replace those with the new page.