Homework 1: Solutions CS 245, Summer 94, Handout 7
Ramkumar July 7, 1994

Problem 1: *(40 points)* We use a uniform definition of giga/mega/kilo byte as being powers of 2 for clarity. For bandwidth, the problem states for information retrieved per track; alternatively, one might assume a parallel read from all tracks in a cylinder, resulting in a factor of 2^4 increase in bandwidth.

- Capacity: $M = 2^{30}$ bytes
- Surfaces: $S = 2^4$
- Cylinders: $C = 2^{10}$
- Rotation: 5400 rpm, $R = 90$ rps, period: $\tau_R = 1/90 = 0.0111$ sec $= 11.1$ ms
- Seek: $\tau_s = 15$ ms
- Overhead: $O = 0.20$
- Tracks: $T = S * C = 2^{14}$

a) The burst bandwidth is the maximum rate of information transfer from the disk. It is $b_b = M/(T\tau_R(1-O)) = 2^{16}/(0.0111 * 0.8)$ bytes/sec $= 1/(16 * 0.0111 * 0.8) = 7.04$ Mb/sec.

b) The sustained bandwidth is the information stored per track divided by the time to retrieve it. It is $b_s = M/(T\tau_R) = 2^{16}/0.0111$ bytes/sec $= 1/(16 * 0.0111) = 5.63$ Mb/sec.

c) The rotational latency is half the rotational period of the disk, which is $\tau_r = \tau_R/2 = 5.56$ ms.

d) The average time to retrieve a "random" 2^{12} byte block is $\tau_b = \tau_s + \tau_r + (2^{12}/b_b) * (1000/2^{20}) = 15 + 5.56 + 0.56 = 21.12$ ms. The multiplication by $(1000/2^{20})$ is used to convert between bytes*sec/Mb and ms.

Problem 2: *(15 points)*

a) The goal of interleaving is that consecutive blocks may be read off or written to the disk immediately, one after the other. A properly designed interleaving ensures that the time between consecutive block requests is equal to the number blocks passed under the disk head. Clearly, it is much better to overestimate the interleaving parameter because if the block request occurs too late, the disk will have to nearly fully rotate before the desired block can be found. Interleaving is commonly found in systems where data transfer cannot overlap with data processing.

b) Double buffering can reduce the time between consecutive block requests, since one buffer is being processed while another is being updated, although that may not necessarily correspond to changing the interleaving. If the time taken to process a buffer exceeds the time to read a block off the disk, interleaving may still prove useful.

c) Interleaving is redundant with infinite memory; ideally, all blocks would fill the buffers and any block request would never wait. This depends on the assumptions made about the disk scheduler/controller and *forced output* – in which case, poor interleavings can cause problems.

Problem 3: *(15 points)*Many answers are acceptable. A few are listed below.

1. Fixed length fields/records are easy to update without rewriting and shifting adjacent data.
2. The record positions are easy to calculate.
3. Fields may be immediately accessed in records without indirection.
4. Fields do not need information describing length and offset.
5. Fixed length records eliminates the need for "compacting" data after insertions and deletions.

Problem 4: *(15 points)*

a) *Move record 6 to space by record 5, record 7 to record 6.* This compacts the space used, preserves the order of the file, though may cause problems with indexing and pinning. It requires 2 record moves in this case (and on average, requires shifting half the records in the file structure).

b) *Move record 7 to record 5.* This compacts space, but does not preserve file order, and may cause problems with indexing and pinning. It requires one record move.

c) *Mark record 5 as deleted.* Does not compact space, is safe with existing indexes and pointers (provided the space is not reclaimed), and preserves order. It requires the setting of a flag or *tombstone* to indicate deletion, and may require updating a *free list* or *file header*.

Problem 5 (extra credit): *(15 points)Since this problem was extra credit, the grading was on how many interesting arguments were made.* The goal of clustering is to locate related data together. It groups records together with common values in a particular field - this can be achieved by sorting. Two different indices may not imply the same clustering unless they are dependent in some way. Hence, for the general case, clustering on two arbitrary indices requires implementing two physically separate data representations - the merits of which for reads/writes, fault-tolerance, and storage, etc. depend on the application.