Algorithms and Data Structures

Ian Munro Editor

The Number of Buffers Required for Sequential Processing of a Disk File

ALON ITAI and YOAV RAZ

ABSTRACT: The number of buffers required for sequential processing of disk files is investigated with the assumption that there is a single user served by two processors: one reads blocks from the disk into buffers in main memory, while the other processor processes the buffers. If processing is faster than reading, then two buffers suffice. However, if processing is slower, using only two buffers does not guarantee minimum completion time. The minimal required number of buffers is calculated in this article.

1. INTRODUCTION

Many programs require a file to be processed consecutively. It is well known that if the reading time of a block is greater than its processing time then two buffers are sufficient to yield the best performance. However, we show that when processing a block requires more time than reading it, two buffers are not always sufficient to ensure minimum completion time. Such situations arise in many practical systems.

We assume that the system consists of two processors working in parallel: a CPU dedicated to processing blocks in buffers and a disk controller dedicated to reading fixed length blocks from the disk.[1] The CPU is not interrupted by the disk controller and there are no other processes competing for the CPU or the disk.

[1] For technical details the author suggests the following publications: *Introduction to IBM Direct Access Storage Devices* by Marilyn Bohl, published by Science Research Associates, Inc. (SR20-4738-00); Reference Manual for IBM 3330 Series Disk Storage (GA26-1615); Reference Manual for IBM 3350 Direct Access Storage (GA26-1638); Introduction to IBM 3380 Direct Action Storage (GA26-1662); IBM 3380 Direct Action Storage Description and User's Guide (GA26-1664); and *Database Design*, 2d. edition by Gio Wiederhold, published by McGraw-Hill, 1983.

We shall make the following assumptions:

(1) Reading and processing can be conducted simultaneously in parallel.
(2) At any instance, no more than one buffer can be processed, and no more than one block can be read into a buffer.
(3) The processing of a block can start only after it has been read into a buffer.
(4) A block may be read into an empty buffer or into a buffer only after the processing of the block currently in the buffer has terminated.

We shall use the following parameters:

R—Reading time of a block
P—Processing time of a block
T—Time of a full rotation of the disk
n—Number of blocks per track,
N—Total number of blocks (of the entire file),
b—Number of available buffers, and
m—Number of tracks (required by the entire file).

Remark 1.1: R is the time the reading head passes over all parts of a block including all intrablock and one interblock gap.

Remark 1.2: P is the time required to process the block by both the system and the application program.

The *completion time*, C, of an algorithm A processing a file F is the entire time required for A to process F, i.e, the time elapsed from starting reading the first block of F until the last block is processed. The *optimal number of buffers* is the minimal number of buffers necessary to process a file within minimum completion time.

Consider the following *Greedy* algorithm which reads the blocks consecutively:

(1) The reader reads the blocks sequentially as long as an empty buffer is available (and not all blocks have been read).
(2) The reader remains idle until a buffer becomes available and the reading head is positioned at the next block to be read.
(3) The processor processes the buffers sequentially emptying the buffer when the processing of a block has terminated.

Let $r(t)$ be the number of blocks the reading of which has started at time t; and $p(t)$ be the number of blocks whose processing has been completed at time t. Then for all $0 \le t \le C$, b, the minimum number of buffers, satisfies:

$$b + p(t) - r(t) \ge 0. \tag{1.1}$$

We find b by considering the values of t for which the value of $r(t) - p(t)$ is maximized and thus derive the minimal number of buffers required.

We shall distinguish between the following cases:

1. $R \le \frac{1}{2}T$.
   1.1 $P \le R$—two buffers are sufficient.
   1.2 $R < P < T$ see below.
   1.3 $T \le P$—two buffers are sufficient.
2. $R > \frac{1}{2}T$ (only one block per track).
   2.1 $P \le T - R$—one buffer is sufficient.
   2.2 $P > T - R$—two buffers are sufficient.

The following analysis deals with case 1.2. Section 2 discusses the case that the file requires no more than a single track. In Section 3 we discuss multi-track files, and conclude with some general remarks.

2. SINGLE TRACK FILES

The following lemma shows that within a track the aforementioned Greedy algorithm is optimal.

LEMMA 1: *Let F be a file residing entirely in a single track of the disk, and A be an algorithm which processes the file sequentially using b buffers (A may read the file in any order). If C^A is the completion time of A and C^G the completion time of the Greedy algorithm when it uses b buffers, then $C^G \le C^A$.*

PROOF: Consider the relationship between R, P and T as analyzed previously. It is clear that the Greedy algorithm is optimal for all cases but case 1.2. The following shows that it is also optimal for case 1.2.

Suppose $C^A < C^G$. For any algorithm $D \ne G$ define $s(D)$ to be the minimum index (≥ 1) such that D reads block $B_{s(D)}$ before reading block $B_{s(d)-1}$. Among all algorithms that use b buffers and whose completion time is C^A, let A' maximize s.

Let $s = s(A')$ and let A'' be the algorithm which reads the blocks in the same sequence as A' except that the reading of B_s is delayed to time t_s—the time A' finished reading B_{s-1}. Since processing is sequential, A' must keep B_{s-1} in memory until after it is processed, i.e., until at least time $t_s + P$. Thus at time t_s the buffer A' used for B_s is available to A''. Since the read head is above B_s at time t_s, A'' can start reading B_s at that time and complete reading it by time $t_s + R \le t_s + P$, the earliest time A' can start processing B_s. Consequently, A' and A'' finish processing B_s at the same time. While A'' read B_s, the read head of A' was idle, thus after time $t_s + R$, A' and A'' are identical. Thus A' and A'' have the same completion time. But $s(A'') \ge s + 1$, contrary to the definition of A'. □

Here we calculate the number of buffers required by the Greedy algorithm to achieve minimum completion time. The only interesting case is case 1.2: If within a track the reader becomes idle it will be idle at least one complete revolution, since this is the time it takes for the next block to return to the read head and become available for reading. Since we assumed $P < T$ at least one buffer has become available during a revolution and reading may resume.

Thus reading should proceed in cycles: filling all available buffers, then a revolution of read idle time, during which at least one buffer is emptied.

Two subcases are possible:

(i) The reader does not become idle until the entire track is read.
(ii) The reader has periods of idle time after which reading is resumed.

Subcase (i): The reader reads all the tracks consecutively. Thus the reader finishes at time NR. During that time there should be enough buffers so that the reader never runs out of empty buffers. Since processing starts after reading the first block, at time $b_1 R$, $(b_1 - 1)R/P$ blocks have already been processed and their buffers have been emptied. At time $bR < t < NR$ the reader has started to read $\lceil t/R \rceil$ blocks. Also $\lfloor (t - R)/P \rfloor$ buffers have been emptied. Thus the number of empty buffers is

$$b_1 - \lceil t/R \rceil + \lfloor (t - R)/P \rfloor \ge 0. \tag{2.1}$$

or

$$b_1 \ge \lceil t/R \rceil - \lfloor (t - R)/P \rfloor.$$

The least number of empty buffers occurs just after starting reading the last block, i.e., slightly after $t = (N - 1)R$. At that time,

$$b_1 = N - \lfloor (N - 2)R/P \rfloor. \tag{2.2}$$

Remark 2.1: Note that when P approaches R from above $\lim_{P \downarrow R} b_1 = 3$.

Subcase (ii): The reader has periods of idle time.

This occurs only if no empty buffer becomes available. In this case it is simpler to use the following argument instead of (1.1).

If the reader has idle time after reading block B_i, block B_{i+1} will not be available in memory until $T + R$ time after completing reading B_i. To prevent the processor from becoming idle the processing of the $b_2 - 1$ available full buffers should require at least $T + R$ time.

For example,

$$(b_2 - 1)P \geq T + R \tag{2.3}$$

or

$$b_2 = 1 + \left\lceil \frac{T + R}{P} \right\rceil. \tag{2.4}$$

Remark 2.2: Note that when P approaches R from above $\lim_{P \downarrow R} b_2 = 3$.

Both subcases (i) and (ii) avoid idle time of the processor. We should choose the subcase which minimizes the number of buffers. Consequently, the number of buffers required by the combined strategy is

$$\begin{aligned} b &= \min\{b_1, b_2\} \\ &= \min\left\{N - \lfloor (N-2)R/P \rfloor, 1 + \left\lceil \frac{T+R}{P} \right\rceil \right\}. \end{aligned} \tag{2.5}$$

Consider the case that the track is full, i.e., $n = N$,

$$T = (N + \theta_1)R \quad \text{where } \theta_1 = \frac{T}{R} - \left\lfloor \frac{T}{R} \right\rfloor \quad (0 \leq \theta_1 < 1).$$

Let $\theta_2 = (N-2)R/P - \lfloor (N-2)R/P \rfloor$ $(0 \leq \theta_2 < 1)$
The number of buffers in subcase (i) is $N - (N-2)R/P + \theta_2$. Subcase (i) occurs when

$$N - (N-2)R/P + \theta_2 \leq 1 + \frac{T+R}{P}$$

$$nP - (N-2)R + \theta_2 P \leq P + T + R = P + (N + \theta_1)R + R$$

$$N(P - 2R) \leq (1 - \theta_2)P - (1 - \theta_1)R.$$

If $P > 2R$ then subcase (i) occurs when

$$N \leq \frac{(1 - \theta_2)P - (1 - \theta_1)R}{(P - 2R)}.$$

When $P < 2R$ the constraint becomes

$$N \geq \frac{P(1 - \theta_2) - (1 - \theta_1)R}{(P - 2R).}$$

Note that there are two crossover points: when $P = 2R$ and when $(1 - \theta_2)P = (1 - \theta_1)R$.

3. READING AN ENTIRE CYLINDER

Let us now consider the general case of a cylinder with m tracks and a total of $N \leq nm$ blocks. When the tracks are full, i.e., $T = nR$ the analysis is identical to that of the previous section. In general, however, $L = T - nR > 0$, i.e., at the end of each track there is some space remaining, of size less than a block, where no data is written. Thus, no data is read while the head passes over that space.

This leftover space causes some anomalies: The first is that the Greedy algorithm is not always optimal. If $P > L + R$, Lemma 1 still holds; but for $R < P < L + R$, we have the following counterexample:

Example 1: Let $b = 2$, $R = 1$, $P = 1.1$, $L = 0.2$, $N = 4$ and $m = 3$. (That is, the first three blocks reside on the first track and the fourth block on the second.)

The Greedy algorithm works as follows: At $t = 1$ B_1 is ready for processing. At $t = 2$ B_2 is also ready, however, B_3 cannot be read since no buffer is available, thus the reader must be idle for an entire revolution $T = 3R + L = 3.2$ by this time both buffers are empty and B_3 is in memory only at time $2 + T + R = 6.2$, and it may be processed. B_4 starts entering at time $6.2 + L = 6.4$ and is in memory at time 7.4. Thus the completion time is 8.5.

The optimal algorithm reads B_4 before B_3 (at the beginning of the second revolution), at that time the first two blocks have already been processed and we have two available buffers, one for B_4 and one for B_3. Thus we have not interfered with the reading and processing of B_3. However, now B_4 is available for processing immediately after the processing of B_3, i.e., at time $6.2 + P = 7.3$. Thus the completion time is $7.3 + P = 8.4$. Somewhat less than the completion time of the Greedy algorithm. □

The second anomaly is that sometimes the processor must be idle (even with an infinite number of buffers): namely, when R/P is sufficiently close to 1, then at the end of a single revolution, n blocks have been read and $(T - R)/P$ blocks have been processed. The last block of the track is processed at time $R + nP$, the next block (the first block of the second track) becomes available at time $T + R$. If $R + nP < T + R$ then the idle time is $T - nP$. Thus the completion time is equal to

the reading time of the first block

+ $(m - 1)$ revolutions

+ the processing time of the last track

$$= R + (m-1)T + (N - (m-1)n)P$$

This occurs when $T - nP > 0$. However, since $T < (n + 1)R$, we get $(n + 1)R - nP > 0$, or $n/(n + 1) < R/P < 1$.

Since the tracks are processed independently, the number of buffers is determined as if there were only a single track, and since while reading a track the reader is not idle, Subcase (i) of Section 2 applies, i.e.,

$$b = n - \lfloor (n-2)R/P \rfloor.$$

Since $n/(n + 1) \leq R/P < 1$ then for $n \geq 3$:

$$b \geq n - \lfloor (n-2) - \epsilon \rfloor = n - (n-3) = 3,$$

and

$$b \leq n - \left\lfloor (n-2)\frac{n}{n+1} \right\rfloor = n - \left\lfloor n - 3 + \frac{3}{n+1} \right\rfloor$$

$$= n - (n-3) = 3.$$

Hence, for $n \geq 3$, $b = 3$.

For $n = 2$, since, $b \geq 2$, we have $b = 2$.

Subcase (i): An entire track is read in each revolution.

Let t be the time since reading began, and $k(t) = \lfloor t/T \rfloor$ the number of tracks already read. At time t the reader has started reading $k(t)n + \lceil (t - k(t)T)/R \rceil$ blocks, and the processing of $\lfloor (t - R)/P \rfloor$ has ended. Thus the number of empty buffers is

$$b - k(t)n - \left\lceil \frac{t - k(t)T}{R} \right\rceil + \left\lfloor \frac{t - R}{P} \right\rfloor \geq 0 \qquad (3.1)$$

or

$$b_1 \geq k(t)n + \left\lceil \frac{t - k(t)T}{R} \right\rceil - \left\lfloor \frac{t - R}{P} \right\rfloor \geq 0. \qquad (3.2)$$

The least number of empty buffers occurs either just after starting to read the last block, or just after starting to read the last block of the $m - 1$st track. In the first case $t = (m-1)T + (N - (m-1)n - 1)R$. Since $k(t) = m - 1$,

$$\left\lceil \frac{t - k(t)T}{R} \right\rceil = \left\lceil \frac{(m-1)T + (N - (m-1)n - 1)R + \epsilon - (m-1)T}{R} \right\rceil$$

$$= N - (m-1)n.$$

$$\left\lfloor \frac{t - R}{P} \right\rfloor = \left\lfloor \frac{(m-1)T + (N - (m-1)n - 1)R + \epsilon - R}{P} \right\rfloor$$

$$= \left\lfloor \frac{(m+1)T + (N - (m-1)n + \epsilon - 2)R}{P} \right\rfloor.$$

Substituting in (3.2)

$$b_1 = (m-1)n + (N - (m-1)n) - \left\lfloor \frac{(m-1)T + (N - (m-1)n - 2)R}{P} \right\rfloor$$

$$= N - \left\lfloor \frac{(m-1)T - (m-1)nR}{P} + \frac{(N-2)R}{P} \right\rfloor$$

$$= N - \left\lfloor \frac{(m-1)(T - nR)}{P} + \frac{(N-2)R}{P} \right\rfloor$$

$$= N - \frac{(m-1)(T - nR)}{P} - \frac{(N-2)R}{P} + \alpha,$$

for some $0 \leq \alpha < 1$.

Note that since $nR \leq T < (n+1)R$, $0 \leq T - nR < R$. The term $(m-1)(T - nR)/P$ comes from the empty space at the end of the track. Since while the head passes over that space, the CPU may finish its processing but no blocks are read, the effect is, as the equation implies, to decrease the number of buffers.

The second case is identical to the first except that the last track plays no part—thus the effective number of blocks is $N' = (m-1)n$ and the number of completed tracks is $m - 2$, thus $m' = m - 1$. Consequently,

$$b_1' = N' - \left\lfloor \frac{(m'-1)(T - nR)}{P} + \frac{(N'-2)R}{P} \right\rfloor$$

$$= (m-1)n - \left\lfloor \frac{(m-2)(T - nR)}{P} + \frac{((m-1)n - 2)R}{P} \right\rfloor.$$

$$= (m-1)n - \frac{(m-2)(T - nR)}{P} - \frac{((m-1)n - 2)R}{P} + \alpha'.$$

Consider the difference between these two expressions:

$$b_1 - b_1' = (N - N')\left(1 - \frac{R}{P}\right) - \frac{T - nR}{P} + \alpha - \alpha'.$$

$$b_1 - b_1' \geq -\frac{T - nR}{P} - \alpha' > -2.$$

However, since the above must be integral,

$$-1 \leq b_1 - b_1' \leq \left\lfloor (N - N')\left(1 - \frac{R}{P}\right) + 1 \right\rfloor.$$

Note that the minimum number of buffers is the maximum of b_1 and b_1'.

Subcase (ii): The reader has idle time before reading is completed.

The analysis here is also similar to that of a single track. The difference is that for a single track the maximum waiting time is $T - R$. Here, we must account for the space between the end of the nth buffer and the end of the track. Therefore, if the reader runs out of buffers just after reading the last buffer of a track the reader must wait $T - nR$ time until getting to the beginning of the next track, so the next buffer becomes available only $R + (T - nR) = T - (n-1)R$ time later. Substituting the latter for R in (2.4) yields

$$b_2 \geq 1 + \left\lceil \frac{2T - (n-2)R}{P} \right\rceil.$$

Note that also here

$$\lim_{P \uparrow T} b = \lim_{P \uparrow T} \left(1 + \left\lceil \frac{2T - (n-2)R}{P} \right\rceil\right)$$

$$= \lim_{\epsilon \downarrow 0} (\lceil 3 - 2\epsilon + (n-2)R/T \rceil) = 3,$$

where $\epsilon = T/P - 1$.

Summary: The following table gives the value of b for more than one track.

	$\frac{n}{n+1} < \frac{R}{P} < 1$	$\frac{R}{P} < \frac{n}{n+1}$
$n = 2$	2	$\min(\max(b_1, b_1'), b_2)$
$n > 2$	3	

See further examples in the appendix to this article.

4. CONCLUSIONS

In practice the Greedy algorithm is always used. We have shown that except for a single pathological case it indeed ensures minimum completion time. If the algorithm is I/O-bound ($R > P$) then two buffers suffice. However, for processing CPU-bound problems in most cases at least three buffers are required to ensure minimum completion time.

We have analyzed the case of a small file which resides on a single track and that of a larger file which occupies several tracks of the same cylinder. We have

not discussed multi-cylinder files, since a cylinder contains a large amount of data and thus its processing time is large. Optimizing over the cylinder boundaries can save at most T, the time for a single revolution, which is much smaller than the processing time of a cylinder. Thus it makes more sense to process each cylinder as a separate file.

For $T > P > [(n + 1)/n]R > 0$, our calculations show that b is a piecewise continuous function of P/R with possible jumps of ± 1. This has practical significance since quite often P is known only approximately and we are forced to use an upper bound for P. The continuity of b ensures that if the estimate of P is not too far from its true value the number of allocated buffers will not significantly exceed the optimal b for the true value of P. Thus we can find an upper bound on the number of buffers and allocate enough buffers to achieve the minimum completion time.

APPENDIX

Examples

Example 1: $R = 1$ $T = 10.5$ $n = 10$ $m = 10$.

The cylinder includes $N = 100$ blocks, each track contains $n = 10$ blocks. (The last track is full, and in the relevant domain $b_1 \geq b_1'$).

If $< P < 1.1$ then $b = 3$ ($R < P < R(n + 1)/n$).

P	b_1	b_2	$b = min(b_1, b_2)$
1.1	7	13	7
1.2	15	12	12
1.3	22	11	11
2	49	8	8
3	66	6	6
4	75	5	5
5	80	4	4
10	90	3	3
10.4	91	3	3

Example 2: $R = 1$ $T = 10.5$ $m = 10$; the number of blocks in the cylinder is $N = 91$; $n = 10$ (however, there is only one block in the last track).

If $1 < P < 1.1$ then $b = 3$.

P	b_1	b_1'	b_2	$b = min(max(b_1, b_1'), b_2)$
1.1	6	7	13	7
1.11	7	8	13	8
1.12	8	8	13	8
1.13	9	9	13	9
1.14	9	10	13	10
1.15	10	10	13	10
1.18	12	13	13	13
1.19	13	13	12	12
1.2	14	14	12	12
1.25	17	17	12	12
1.3	20	20	11	11

CR Categories and Subject Descriptors: B.3.2 [**Memory Structures**]: Design Styles—*sequential-access memory*; C.4 [**Performance of Systems**]: D.4.4 [**Communications Management**]: Buffering

Additional Key Words and Phrases: Buffering, disk files, performance evaluation, sequential files

ABOUT THE AUTHORS:

ALON ITAI is an associate professor at the Computer Science Department of the Technion–Israel Institute of Technology, Haifa. He has done research in data structures, probabilistic analysis of algorithms, and distributed algorithms. Author's present address: Alon Itai, Computer Science Department, Technion–Israel Institute of Technology, Haifa, Israel.

YOAV RAZ is currently with the OLTP Systems Architecture Group, Digital Equipment Corporation. He was cofounder and managing director of Sitav Ltd. (a software house), Israel; senior lecturer at the Computer Science Department of the Technion–Israel Institute of Technology, Haifa; and a visiting associate professor at the Computer Science and Engineering Department, University of California at San Diego. His research interests include database management systems and theory of computation. Author's present address: Yoav Raz, Digital Equipment Corporation, 200 Forest Street, MR01-1/A65, Marlboro, MA 01752.