## HARD DISKS

### 5 DISK STRUCTURE

<table>
<thead>
<tr>
<th>Optimizing disk arm movements total movement and algorithm complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>6 ORDERING TRACK REQUEST FOR MINIMUM ARM TRAVEL</strong></td>
</tr>
<tr>
<td>Comparisons Of SSF and Elevator 1</td>
</tr>
<tr>
<td><strong>7 ELEVATOR ALGORITHMS WORST CASE WAITS</strong></td>
</tr>
<tr>
<td>ARM MOVEMENT ANALYSIS -SSF and ELEVATOR 1 Problems</td>
</tr>
<tr>
<td><strong>8 IMPLEMENTING FCFS AND SSF AND ELEVATOR</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimizing disk arm movements total movement and algorithm complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>9 THE NEED FOR BUFFERING</strong></td>
</tr>
<tr>
<td><strong>10 SINGLE AND DOUBLE BUFFERING DISK---&gt;MM</strong></td>
</tr>
<tr>
<td>DISK CONTROLLER FILLS DMA or DISK CONTROLLER EMPTIES</td>
</tr>
<tr>
<td><strong>11 GENERAL BUFFERING 1-LEVEL</strong></td>
</tr>
<tr>
<td><strong>12 GENERAL BUFFERING 2-LEVEL</strong></td>
</tr>
<tr>
<td><strong>13 MORE ON BUFFERS-TME TO FILL AND TIME TO EMPTY</strong></td>
</tr>
<tr>
<td>Interleaving-Uniformity By relation of number of sectors and interleaving factor</td>
</tr>
<tr>
<td><strong>14 BUFFERING AND INTERLEAVING</strong></td>
</tr>
<tr>
<td><strong>15 INTERLEAVING-COLLISIONS</strong></td>
</tr>
<tr>
<td><strong>16 INTERLEAVING-JUSTIFICATION OF GCD, LCM</strong></td>
</tr>
<tr>
<td><strong>17 BUFFERING AND LITTLES LAW GENERAL</strong></td>
</tr>
<tr>
<td><strong>18 DYNAMIC ALLOCATION BUFFERING</strong></td>
</tr>
</tbody>
</table>
BACKGROUND HARDWARE: TOPIC: INPUT-OUTPUT

DISK (driver) block, sectors, tracks, cylinder, (controller) move arms, heads, detect block#

interrupt or trap - on which instruction?

Simple

Pipelined

SuperScalar

Fetch Decode Ex

Fetch Decode Ex

Fetch Decode Ex

Fetch Decode Ex

CPU

ALU REGs

IR

Decode

CONTROL

MM

Main Mem

RAM

DMA

With DMA MM buffer must not be swapped out

Op Sys

Instr Reg GpRegs

PrgCntr StckPtr

PrgStatWd

Cache 1

Cache 2

BRIDGE

Interrupt test after each machine instr.

User I/O -> Drivers -> Controller -> Device

page address is not in associative mem - go to page table page is not in MM - page fault OS must get page in VM

User I/O -> Drivers -> Controller -> Device

Disk Controller

Interrupt Controller

Printer Controller

Video RAM

Video Controller

Graphics Adapter

Monitor

© M C Paull 2005
The CPU initiates IO by giving basic requirements to a controller. The controller then carries directs IO device independent of CPU. When complete sends interrupt.

CPU issues Bus Order to Read from Disk to Disk Controller Buffer. When Buffer Full interrupt occurs CPU orders DMA to Start Controlling Transfer to MM [1 word at a time, or a series of words—burst mode.]

Conflict between Disk Controller and CPU goes to Disk Controller—Cycle Stealing.

Separate Connections
CPU Controlled

The CPU controls IO data transfer directly. Each IO device is given an MM type address. The CPU refers to IO by that address sent on the same bus as MM.

Programmed IO

interrupt DMA IO

Interrupt IO

Shared Bus Communication

The CPU controls IO data transfer directly. Each IO device is given an MM type address. The CPU refers to IO by that address sent on the same bus as MM.
WRITE: PROGRAMMED IO, INTERRUPT IO, DMA IO
ARM MOVEMENT ALGORITHMS

I  FCFS   First Come First Served
II SSF   Shortest Seek First
III ELEVATOR 1 Pick up all request, as head move from hub to rim and also as it moves from rim to hub, etc. 2 Pick up all request, as head moves from hub to rim then go non-pickup to hub.

BUFFERS
With Multiprogramming Disk Block Requests, as well as Keystrokes, etc., can arrive faster than they can be handled—so they are Buffered (Stored until Controller is ready) This can be in a way which depends on the Arm Movement Alg.
Arm Movement Is Proportional To Time Spent Moving Assuming Average Speed Is the Same.

**ORDERING TRACK REQUEST FOR MINIMUM ARM TRAVEL**
Comparisons Of SSF aand Elevator 1

© M.C. Paull
The disk arm is at position \( p \) left of midway (Same if \( p \) is right of midway).

The arm receives \( n \) calls from areas \( U, U' \) and \( X \). \( U \) is the distance to outer edge. \( U' \) is equal in extent to \( U \). The SSF algorithm moves the arm to the closest call first. What is the probability that it will move in either direction?

**Simple Analysis:** The probability that all arm calls are in \( X \) is \( g^n \), if so the arm moves toward Midway. Alternatively, if there are calls in \( U \) or \( U' \) it is equally likely that the closest to \( p \) is in \( U \) as in \( U' \). So the difference in probability that \( p \) will move in either direction favors “toward Midway” with a probability difference of \( g^n \). The analysis assumes that each track is equally likely to hold requests. But the tendency to move toward Midway means that requests near the hub and outer edge will tend to be unserved and so accumulate. This will moderate the tendency to move toward Midway—but it will not eliminate it. This tendency is further moderated in modern disks in which track density increases toward the outer edge.

Shortest Seek First (SSF) Algorithm FAVORS MIDWAY

**ELEVATOR ALGORITHMS WORST CASE WAITS**

**ARM MOVEMENT ANALYSIS -SSF and ELEVATOR 1 Problems**

©M.C. Paull
IMPLEMENTING FCFS-Circular Buffer, SSF AND ELEVATOR-Linked List ALGORITHMS
Buffering Situations: General: From Producer to Consumer

1. Mismatch in quantity needed for efficient operation
   a) Producers Optimal Unit size > Consumer Optimal Unit => Consumer must stop or do something else.
   b) Producers Optimal Unit size < Producer Optimal Unit => Consumer Rate or do something else.

1. Mismatch in speed between Producers and Consumer of Data
   a) For data D1 Producers Rate > Consumer Rate => Producer must stop or do something else.
   b) For data D2 Producers Rate < Consumer Rate => Consumer must stop or do something else.

Buffering Situations: Specific OS

From Disk to MM, From MM to Disk
From Modem to Disk: Buffered in MM until enough for a write to disk in one operation is accumulated
From Keyboard to Screen or to Application

Relative Speeds

<table>
<thead>
<tr>
<th>Devices</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>10^1</td>
<td>10^2</td>
<td>10^3</td>
<td>10^4</td>
<td>10^5</td>
</tr>
<tr>
<td>SBUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fast</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ethernet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hard disk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ethernet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>laser</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>printer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>modem</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>keyboard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relative Speeds: Silberschaz, et al
The Rotating Disk will allow Sectors to Pass Unread while the Disk Buffer is Unloaded.
Double Buffering - DEVICE -> CONTROLLER -> PROCESS
Emptying and filling overlap

Single Buffering - DEVICE-CONTROLLER-PROCESS
Emptying and filling are disjoint

Emptying on even
Emptying on odd

6, 8
2, 4

5, 7
3, 5

t =

1, 3

cb1

5, 7 enter
on odd

6, 8 enter
on even

t =

cb2

GENERAL BUFFERING 1 LEVEL

© M.C. Paull
DEVISE

DEVICE---CONTROLLED---DRIVER->PROCESS

Double Buffering

GENERAL BUFFERING 2 LEVEL
Double Buffering: Can only fill 1 buffer at a time, so filling time, $n$, dominates.

Time to do $M$ is $Mn$

Single Buffering

Time to do $M$ is $M(n + xn) = Mn + Mnx$

Time to Fill Buffer >= Time to Empty Buffer ($x <or= n$)

Double Buffering can only empty 1 buffer at a time, so emptying time, $xn$, dominates.

(after 1st) Time to do $M$ is $Mxn$

Time to Fill Buffer < Time to Empty Buffer ($x >or= 1$)

Cannot Be Emptying Both Buffers At the same Time

$n$ = Time to Fill
$xn$ = Time to Empty
$M$ = Number of Buffer Fill and Empty cycles

MORE ON BUFFERS-TIME TO FILL AND TIME TO EMPTY

© M.C. Paull 13
In Single (1) Interleaving The Rotating Disk will allow 1 Sectors to Pass Unread while the Disk Buffer is Unloaded So Files can be occupy every other or every kth segment so the loading of a file can be done continuously.

In Single (k) Interleaving The Rotating Disk will allow k Sectors to Pass Unread while the Disk Buffer is Unloaded So Files can be occupy every every kth segment so the loading of a file can be done continuously.

SINGLE BUFFERING and INTERLEAVING (Empty > Filling Time (<?))

© M.C. Paull
If there are \( N \) sectors and they are interleaved so that they are chosen at *Intervals* of \( J \) [\( \text{Interval} = \# \text{of segments counting from beginning of one to beginning of } J \text{th} \) then:

If there are \( N \) segments chosen before segment 0 is re-chosen there is no collision

If there are \( <N \) segments chosen before a return to 0 then there is a collision.

[If we start at any sector and after \( k \) Js return to that sector then if we start at 0 after \( k \) Js we will return to 0-so we use 0 as the starting point]

\[
J = \text{Interval} = \# \text{of segments counting from first to beginning of } J \text{th segments to be read}
\]

\( N \) is number of segments from start of segment 0 to next start segment 0=total # of segments

\( qN \) with \( q = 1, 2, \ldots \) is the positions successive of 0s

\( pJ \) with \( p = 1, 2, \ldots \) is the position of successive reads.

In general there is a collision iff the smallest positive integers \( q < J \) and \( p < N \) such that \( qN = pxJ \).

An equivalent statement is:

iff \( N \) and \( J \) are relatively prime i.e they have no common divisor other than 1, there will be no collision.

INTERLEAVING-EXAMPLE JUSTIFICATION
In general there is collision if there are positive integers \( q \) and \( p \) such that

\[ qxN = pxJ \text{ and } q < J \]

\( N \) and \( J \) can be represented as products of powers of primes.

\[ N = [p_1^{a_1} p_2^{a_2} \ldots p_n^{a_n}] \quad J = [q_1^{b_1} q_2^{b_2} \ldots q_m^{b_m}] \]

So—in general there is a collision if there are integers \( p \) and \( q \) such that

\[ q [p_1^{a_1} p_2^{a_2} \ldots p_n^{a_n}] = p[q_1^{b_1} q_2^{b_2} \ldots q_m^{b_m}] \quad \ldots \ldots (1) \]

and \( q < J \)

This can not happen if there is no \( p_j = q_k \), for any \( j \) and \( k \). If however if \( q_j = p_k \), assume in particular that \( p_1 = q_1 \), then

\[ \frac{[q_1^{b_1} q_2^{b_2} \ldots q_m^{b_m}]}{p_1} \frac{[p_1^{a_1} p_2^{a_2} \ldots p_n^{a_n}]}{q_1} = \frac{[p_1^{a_1} p_2^{a_2} \ldots p_n^{a_n}]}{q_1} [q_1^{b_1} q_2^{b_2} \ldots q_m^{b_m}]) \]

Let \( q = [q_1^{b_1} q_2^{b_2} \ldots q_m^{b_m}] \), \( p = [p_1^{a_1} p_2^{a_2} \ldots p_n^{a_n}] \)

then \( qxN = pxJ \) and \( q < J \)

In general then if \( N \) and \( J \) are both divisible by the same number then sectors on an \( N \) sector disk will not all be chosen if the interval between successive chosen sectors is \( I = J - 1 \). That is they will not all be chosen unless \( gcd(N,J) = 1 \), or alternatively unless \( lcm(N,J) = N \times J \).
Buffering Is Of Use When Rate Capabilities of Producer and Consumer are Different and/or when Rate Capabilities Of Producer and Consumer Vary Over Time.

Rate Capabilities (in Fraction of Buffer/Unit Time)
Producer can put \( R_{prod} \) in (buffer- fractions/unit time)
\[ \text{ex.: (333 a buffer/msec)} \]
Consumers can remove \( R_{cons} \) in (buffer- fractions/unit time)

\[ \text{Ave-Rprod} = \text{Ave-Fraction Fill-Of-Buffer/Ave -(Waiting) Time-In-Buffer} = \text{Ave-Rcons} \]

[ Little’s Steady State Formula: \( \lambda = n / W \) ]
(other use: Ready Queue-Concern: Waiting Time-Action:Vary number of processes in MM)

**MODES**

Mode 1 (Continuous Stop When Full/Empty)
Prod-Con, OS Schedule Buffer

Producer fills fraction \( f_i \) then Consumer empties fraction \( g_i \) of Buffer--
Repeat with the fractions changing

Mode 1 (Blocking)

The maintenance of an equal average rate of filling and emptying by Producer and is maintained by synchronization-stopping the Producer from filling when the Buffer is full, and stopping the Consumer from emptying when the Buffer is empty. Stopping may be implemented by blocking, busy waiting, sleeping or exiting.

Assume Producer need never lose any input because buffer is full nor Consumer wait because buffer is empty. If \( F_{f.ave} \) is the ave fraction of the Buffer which is filled in a time \( T \) and \( F_{e.ave} \) is the ave fraction of the Buffer which is emptied in a time \( T \). Then the ave rate at which a buffer is filled = \( F_{f.ave} /T \)
and emptied = \( F_{e.ave} /T \) and these must be equal.

Mode 2 (Alternating)
Disk, Modem Single-Double Buffering

Producer fills entire Buffer then Consumer empties entire Buffer-
Repeat-Always entire Buffer

The equal average rate of filling and emptying is forced in this mode by continuing filling (with no emptying) until a buffer is full, and stopping any filling until it has been completely emptied. In single buffering: If \( T_F \) is the total time to fill, and \( T_E \) the time to empty the buffer, then given that they cannot both be done at the same time the average rate of filling = \( 1/(T_F + T_E) \) = time of emptying.

For double buffering, given, as for single buffering that \( T \) is the time to fill and \( T \) the time to empty, since these can occur simultaneously the rate of filling is \( 1/T \).

**HOW THE RATE IN AND OUT OF BUFFERS IS REGULATED SO AS TO GUARANTEE**
**THERE IS NO LOSS AND AVE RATE IN = AVE RATE OUT.**
It only can be done if the regulations are feasible

**BUFFERING AND LITTLE'S LAW GENERAL**
 Buffered Key Strokes With Many Terminals
(Similar To Long File Name Arrangement )

DYNAMIC ALLOCATION BUFFERING