I/O Management and Disk Scheduling

Chapter 11
Categories of I/O Devices

• Human readable
  – Used to communicate with the user
  – Printers
  – Video display terminals
    • Display
    • Keyboard
    • Mouse
Categories of I/O Devices

• Machine readable
  – Used to communicate with electronic equipment
  – Disk and tape drives
  – Sensors
  – Controllers
  – Actuators
Categories of I/O Devices

• Communication
  – Used to communicate with remote devices
  – Digital line drivers
  – Modems
Differences in I/O Devices

- Data rate
  - May be differences of several orders of magnitude between the data transfer rates
Figure 11.1 Typical I/O Device Data Rates
Differences in I/O Devices

• Application
  – Disk used to store files requires file management software
  – Disk used to store virtual memory pages needs special hardware and software to support it
  – Terminal used by system administrator may have a higher priority
Differences in I/O Devices

• Complexity of control

• Unit of transfer
  – Data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk

• Data representation
  – Encoding schemes

• Error conditions
  – Devices respond to errors differently
Performing I/O

• Programmed I/O
  – Process is busy-waiting for the operation to complete

• Interrupt-driven I/O
  – I/O command is issued
  – Processor continues executing instructions
  – I/O module sends an interrupt when done
Performing I/O

- Direct Memory Access (DMA)
  - DMA module controls exchange of data between main memory and the I/O device
  - Processor interrupted only after entire block has been transferred
# Relationship Among Techniques

## Table 11.1 I/O Techniques

<table>
<thead>
<tr>
<th></th>
<th>No Interrupts</th>
<th>Use of Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O-to-memory transfer</td>
<td>Programmed I/O</td>
<td>Interrupt-driven I/O</td>
</tr>
<tr>
<td>through processor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct I/O-to-memory</td>
<td></td>
<td>Direct memory access (DMA)</td>
</tr>
<tr>
<td>transfer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Evolution of the I/O Function

- Processor directly controls a peripheral device
- Controller or I/O module is added
  - Processor uses programmed I/O without interrupts
  - Processor does not need to handle details of external devices
Evolution of the I/O Function

• Controller or I/O module with interrupts
  – Processor does not spend time waiting for an I/O operation to be performed

• Direct Memory Access
  – Blocks of data are moved into memory without involving the processor
  – Processor involved at beginning and end only
Evolution of the I/O Function

• I/O module is a separate processor
• I/O processor
  – I/O module has its own local memory
  – It's a computer in its own right
Direct Memory Access

- Processor delegates I/O operation to the DMA module
- DMA module transfers data directly to or from memory
- When complete DMA module sends an interrupt signal to the processor
DMA

Figure 11.2 Typical DMA Block Diagram
DMA Configurations

(a) Single-bus, detached DMA

(b) Single-bus, Integrated DMA-I/O
DMA Configurations

Figure 11.3  Alternative DMA Configurations
Operating System Design

Issues

• Efficiency
  – Most I/O devices extremely slow compared to main memory
  – Use of multiprogramming allows for some processes to be waiting on I/O while another process executes
  – I/O cannot keep up with processor speed
  – Swapping is used to bring in additional Ready processes which is an I/O operation
Operating System Design
Issues

• Generality
  – Desirable to handle all I/O devices in a uniform manner
  – Hide most of the details of device I/O in lower-level routines so that processes and upper levels see devices in general terms such as read, write, open, close, lock, unlock
Figure 11.4  A Model of I/O Organization
I/O Buffering

• Reasons for buffering
  – Processes must wait for I/O to complete before proceeding
  – Certain pages must remain in main memory during I/O
I/O Buffering

• Block-oriented
  – Information is stored in fixed sized blocks
  – Transfers are made a block at a time
  – Used for disks and tapes

• Stream-oriented
  – Transfer information as a stream of bytes
  – Used for terminals, printers, communication ports, mouse and other pointing devices, and most other devices that are not secondary storage
Single Buffer

- Operating system assigns a buffer in main memory for an I/O request
- Block-oriented
  - Input transfers made to buffer
  - Block moved to user space when needed
  - Another block is moved into the buffer
- Read ahead
Single Buffer

- **Block-oriented**
  - User process can process one block of data while next block is read in
  - Swapping can occur since input is taking place in system memory, not user memory
  - Operating system keeps track of assignment of system buffers to user processes
Single Buffer

- **Stream-oriented**
  - Used a line at time
  - User input from a terminal is one line at a time with carriage return signaling the end of the line
  - Output to the terminal is one line at a time
I/O Buffering

(a) No buffering

(b) Single buffering
Double Buffer

- Use two system buffers instead of one
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer

(c) Double buffering
Circular Buffer

• More than two buffers are used
• Each individual buffer is one unit in a circular buffer
• Used when I/O operation must keep up with process
Disk Performance Parameters

- To read or write, the disk head must be positioned at the desired track and at the beginning of the desired sector
- Seek time
  - Time it takes to position the head at the desired track
- Rotational delay or rotational latency
  - Time its takes for the beginning of the sector to reach the head
Timing of a Disk I/O Transfer

![Diagram showing the timing of a Disk I/O Transfer](image)

Figure 11.6 Timing of a Disk I/O Transfer
Disk Performance Parameters

• Access time
  – Sum of seek time and rotational delay
  – The time it takes to get in position to read or write

• Data transfer occurs as the sector moves under the head
Seek time is the reason for differences in performance.
For a single disk there will be a number of I/O requests.
If requests are selected randomly, we will have poor performance.
Disk Scheduling Policies

- First-in, first-out (FIFO)
  - Process request sequentially
  - Fair to all processes
  - Approaches random scheduling in performance if there are many processes
Disk Scheduling Policies

• Priority
  – Goal is not to optimize disk use but to meet other objectives
  – Short batch jobs may have higher priority
  – Provide good interactive response time
Disk Scheduling Policies

• Last-in, first-out
  – Good for transaction processing systems
    • The device is given to the most recent user so there should be little arm movement
  – Possibility of starvation since a job may never regain the head of the line
Disk Scheduling Policies

• Shortest Service Time First
  – Select the disk I/O request that requires the least movement of the disk arm from its current position
  – Always choose the minimum Seek time
Disk Scheduling Policies

- **SCAN**
  - Arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction
  - Direction is reversed
Disk Scheduling Policies

• C-SCAN
  – Restricts scanning to one direction only
  – When the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again
Disk Scheduling Policies

- **N-step-SCAN**
  - Segments the disk request queue into subqueues of length N
  - Subqueues are processed one at a time, using SCAN
  - New requests added to other queue when queue is processed

- **FSCAN**
  - Two queues
  - One queue is empty for new requests
## Disk Scheduling Algorithms

### Table 11.2 Comparison of Disk Scheduling Algorithms

<table>
<thead>
<tr>
<th></th>
<th>(a) FIFO (starting at track 100)</th>
<th>(b) SSTF (starting at track 100)</th>
<th>(c) SCAN (starting at track 100, in the direction of increasing track number)</th>
<th>(d) C-SCAN (starting at track 100, in the direction of increasing track number)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next track accessed</td>
<td>Number of tracks traversed</td>
<td>Next track accessed</td>
<td>Number of tracks traversed</td>
<td>Next track accessed</td>
</tr>
<tr>
<td>55</td>
<td>45</td>
<td>90</td>
<td>10</td>
<td>150</td>
</tr>
<tr>
<td>58</td>
<td>3</td>
<td>58</td>
<td>32</td>
<td>160</td>
</tr>
<tr>
<td>39</td>
<td>19</td>
<td>55</td>
<td>3</td>
<td>184</td>
</tr>
<tr>
<td>18</td>
<td>21</td>
<td>39</td>
<td>16</td>
<td>90</td>
</tr>
<tr>
<td>90</td>
<td>72</td>
<td>38</td>
<td>1</td>
<td>58</td>
</tr>
<tr>
<td>160</td>
<td>70</td>
<td>18</td>
<td>20</td>
<td>55</td>
</tr>
<tr>
<td>150</td>
<td>10</td>
<td>150</td>
<td>132</td>
<td>39</td>
</tr>
<tr>
<td>38</td>
<td>112</td>
<td>160</td>
<td>10</td>
<td>38</td>
</tr>
<tr>
<td>184</td>
<td>146</td>
<td>184</td>
<td>24</td>
<td>18</td>
</tr>
<tr>
<td><strong>Average seek length</strong></td>
<td><strong>55.3</strong></td>
<td><strong>Average seek length</strong></td>
<td><strong>27.5</strong></td>
<td><strong>Average seek length</strong></td>
</tr>
</tbody>
</table>
RAID

• Redundant Array of Independent Disks
• Set of physical disk drives viewed by the operating system as a single logical drive
• Data are distributed across the physical drives of an array
• Redundant disk capacity is used to store parity information
RAID 0 (non-redundant)
RAID 1 (mirrored)
RAID 2 (redundancy through Hamming code)
RAID 3 (bit-interleaved parity)
RAID 4 (block-level parity)
RAID 5 (block-level distributed parity)
RAID 6 (dual redundancy)
Disk Cache

- Buffer in main memory for disk sectors
- Contains a copy of some of the sectors on the disk
Least Recently Used

- The block that has been in the cache the longest with no reference to it is replaced
- The cache consists of a stack of blocks
- Most recently referenced block is on the top of the stack
- When a block is referenced or brought into the cache, it is placed on the top of the stack
Least Recently Used

- The block on the bottom of the stack is removed when a new block is brought in.
- Blocks don’t actually move around in main memory.
- A stack of pointers is used.
Least Frequently Used

- The block that has experienced the fewest references is replaced
- A counter is associated with each block
- Counter is incremented each time block accessed
- Block with smallest count is selected for replacement
- Some blocks may be referenced many times in a short period of time and the reference count is misleading
Figure 11.9 Frequency-Based Replacement
Figure 11.10 Some Disk Cache Performance Results Using LRU
Figure 11.11 Disk Cache Performance Using Frequency-Based Replacement [ROBI90]
UNIX SCR4 I/O

- Each individual device is associated with a special file
- Two types of I/O
  - Buffered
  - Unbuffered

Figure 11.12  UNIX I/O Structure
Figure 11.13 UNIX Buffer Cache Organization
Linux I/O

• Elevator scheduler
  – Maintains a single queue for disk read and write requests
  – Keeps list of requests sorted by block number
  – Drive moves in a single direction to satisfy each request
Linux I/O

• Deadline scheduler
  – Uses three queues
    • Incoming requests
    • Read requests go to the tail of a FIFO queue
    • Write requests go to the tail of a FIFO queue
  – Each request has an expiration time
Linux I/O

Figure 11.14  The Linux Deadline I/O Scheduler
Linux I/O

- Anticipatory I/O scheduler
  - Delay a short period of time after satisfying a read request to see if a new nearby request can be made
Windows I/O

• Basic I/O modules
  – Cache manager
  – File system drivers
  – Network drivers
  – Hardware device drivers
Windows I/O

Figure 11.15  Windows I/O Manager