Time(workload) = Time(CPU) + Time(I/O) - Time(Overlap)
I/O Controller Architecture

Peripheral Bus (VME, FutureBus, etc.)

- Host Memory
- Processor Cache
- Host Processor

- Peripheral Bus Interface/DMA
- Buffer Memory
- ROM
- I/O Channel Interface
- μProc

Request/response block interface
Backdoor access to host memory
I/O: A System Performance Perspective

• CPU Performance: Improvement of 60% per year.

• I/O Sub-System Performance: Limited by *mechanical* delays (disk I/O). Improvement less than 10% per year (IO rate per sec or MB per sec).

• From Amdahl's Law: overall system speed-up is limited by the slowest component:

  If I/O is 10% of current processing time:
  • Increasing CPU performance by 10 times
    ⇒ 5 times system performance increase
    (50% loss in performance)
  • Increasing CPU performance by 100 times
    ⇒ 10 times system performance
    (90% loss of performance)

• The I/O system performance bottleneck diminishes the benefit of faster CPUs on overall system performance.
Magnetic Disks

**Characteristics:**

- Diameter: 2.5in - 5.25in
- Rotational speed: 3,600 RPM - 10,000 RPM
- Tracks per surface.
- Sectors per track: Outer tracks contain more sectors.
- Recording or Areal Density: Tracks/in × Bits/in
- Cost Per Megabyte.
- Seek Time: The time needed to move the read/write head arm.
  - Reported values: Minimum, Maximum, Average.
- Rotation Latency or Delay:
  - The time for the requested sector to be under the read/write head.
- Transfer time: The time needed to transfer a sector of bits.
- Type of controller/interface: SCSI, EIDE
- Disk Controller delay or time.
- Average time to access a sector of data =
  
  \[
  \text{average seek time} + \text{average rotational delay} + \text{transfer time} + \text{disk controller overhead}
  \]
Cost Vs. Access Time for:
SRAM, DRAM, Magnetic Disk
Magnetic Disk Cost Vs. Time
The price per megabyte of disk storage has been decreasing at about 40% per year based on improvements in data density, even faster than the price decline for flash memory chips. Recent trends in HDD price per megabyte show an even steeper reduction.
Since the 1980's smaller form factor disk drives have grown in storage capacity. Today's 3.5 inch form factor drives designed for the entry-server market can store more than 75 Gbytes at the 1.6 inch height on 5 disks.
Drive areal density has increased by a factor of 8.5 million since the first disk drive, IBM's RAMAC, was introduced in 1957. Since 1991, the rate of increase in areal density has accelerated to 60% per year, and since 1997 this rate has further accelerated to an incredible 100% per year.
IBM HDD Access/Seek Time-Performance Increase

- Accessing
- Rotating
- Seeking

seek time \( \sim \frac{\text{inertia/\text{power}}}{3} \times \text{(data band)}^{2/3} \)
rotational time \( \sim (\text{RPM})^{-1} \)

IBM Advanced Technology

Availability Year
## IBM HDD Design Projections

<table>
<thead>
<tr>
<th></th>
<th>1999</th>
<th>2000</th>
<th>2002</th>
<th>2004</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5 Inch HDD</td>
<td>36 GB</td>
<td>36 GB</td>
<td>36 GB</td>
<td></td>
</tr>
<tr>
<td>41.3 mm high</td>
<td>10000 RPM</td>
<td>10000 RPM</td>
<td>15000 RPM</td>
<td></td>
</tr>
<tr>
<td>25.4 mm</td>
<td>5.4 ms Tseek</td>
<td>4.9 ms Tseek</td>
<td>4.0 ms Tseek</td>
<td></td>
</tr>
<tr>
<td><strong>Entry-Server (Desktop)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5 Inch HDD</td>
<td>37 GB</td>
<td>75 GB</td>
<td>150 GB</td>
<td>150 GB</td>
</tr>
<tr>
<td>25.4 mm high</td>
<td>7200 RPM</td>
<td>7200 RPM</td>
<td>10000 RPM</td>
<td>15000 RPM</td>
</tr>
<tr>
<td>17 mm</td>
<td>8.5 ms Tseek</td>
<td>8.5 ms Tseek</td>
<td>7.0 ms Tseek</td>
<td>4.0 ms Tseek</td>
</tr>
<tr>
<td><strong>Mobile</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5 Inch HDD</td>
<td>25 GB</td>
<td>32 GB</td>
<td>50 GB</td>
<td>100 GB</td>
</tr>
<tr>
<td>17 mm high</td>
<td>5400 RPM</td>
<td>5400 RPM</td>
<td>5400 RPM</td>
<td>7200 RPM</td>
</tr>
<tr>
<td>12.5 mm</td>
<td>12 ms Tseek</td>
<td>12 ms Tseek</td>
<td>10 ms Tseek</td>
<td>8.0 ms Tseek</td>
</tr>
<tr>
<td><strong>Consumer</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0 Inch HDD</td>
<td>0.34 GB</td>
<td>1.0 GB</td>
<td>1.3 GB</td>
<td>3.0 GB</td>
</tr>
<tr>
<td>5.0 mm high</td>
<td>3600 RPM</td>
<td>3600 RPM</td>
<td>3600 RPM</td>
<td>4500 RPM</td>
</tr>
<tr>
<td></td>
<td>15 ms Tseek</td>
<td>12 ms Tseek</td>
<td>12 ms Tseek</td>
<td>10 ms Tseek</td>
</tr>
</tbody>
</table>
Disk Access Time Example

• Given the following Disk Parameters:
  – Transfer size is 8K bytes
  – Advertised average seek is 12 ms
  – Disk spins at 7200 RPM
  – Transfer rate is 4 MB/sec
• Controller overhead is 2 ms
• Assume that the disk is idle, so no queuing delay exist.
• What is Average Disk Access Time for a 512-byte Sector?
  – Ave. seek + ave. rot delay + transfer time + controller overhead
  – 12 ms + 0.5/(7200 RPM/60) + 8 KB/4 MB/s + 2 ms
  – 12 + 4.15 + 2 + 2 = 20 ms
• Advertised seek time assumes no locality: typically 1/4 to 1/3 advertised seek time: 20 ms => 12 ms
I/O Connection Structure

Different computer system architectures use different degrees of separation between I/O data transmission and memory transmissions.

- **Isolated I/O:** Separate memory and I/O buses
  - A set of I/O device address, data and control lines form a separate I/O bus.
  - Special input and output instructions are used to handle I/O operations.

- **Shared I/O:**
  - Address and data wires are shared between I/O and memory buses.
  - Different control lines for I/O control.
  - Different I/O instructions.

- **Memory-mapped I/O:**
  - Shared address, data, and control lines for memory and I/O.
  - Data transfer to/from the CPU is standardized.
  - Common in modern processor design; reduces CPU chip connections.
  - A range of memory addresses is reserved for I/O registers.
  - I/O registers read/written using standard load/store instructions.
Typical CPU-Memory and I/O Bus Interface
I/O Interface

I/O Interface, controller or I/O bus adapter:

- Specific to each type of I/O device.
- To the CPU, and I/O device, it consists of a set of control and data registers within the I/O address space.
- On the I/O device side, it forms a localized I/O bus which can be shared by several I/O devices.
- Handles I/O details such as:
  - Assembling bits into words,
  - Low-level error detection and correction
  - Accepting or providing words in word-sized I/O registers.
  - Presents a uniform interface to the CPU regardless of I/O device.
Types of Buses

• Processor-Memory Bus (sometimes also called Backplane Bus):
  – Offers very high-speed and low latency.
  – Matched to the memory system to maximize memory-processor bandwidth.
  – Usually design-specific, though some designs use standard backplane buses.

• I/O buses (sometimes called a channel):
  – Follow bus standards.
  – Usually formed by I/O interface adapters to handle many types of connected I/O devices.
  – Wide range in the data bandwidth and latency
  – Not usually interfaced directly to memory but use a processor-memory or backplane bus.
  – Examples: Sun’s SBus, Intel’s PCI, SCSI.
# Main Bus Characteristics

<table>
<thead>
<tr>
<th>Option</th>
<th>High performance</th>
<th>Low cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus width</td>
<td>Separate address &amp; data lines</td>
<td>Multiplex address &amp; data lines</td>
</tr>
<tr>
<td>Data width</td>
<td>Wider is faster (e.g., 32 bits)</td>
<td>Narrower is cheaper (e.g., 8 bits)</td>
</tr>
<tr>
<td>Transfer size</td>
<td>Multiple words has less bus overhead</td>
<td>Single-word transfer is simpler</td>
</tr>
<tr>
<td>Bus masters</td>
<td>Multiple (requires arbitration)</td>
<td>Single master (no arbitration)</td>
</tr>
<tr>
<td>Split</td>
<td>Yes, separate Request and Reply packets gets higher bandwidth (needs multiple masters)</td>
<td>No, continuous transaction? connection is cheaper and has lower latency</td>
</tr>
<tr>
<td>Clocking</td>
<td>Synchronous</td>
<td>Asynchronous</td>
</tr>
</tbody>
</table>
Typical Bus Read Transaction

- Synchronous bus example
- The read begins when the read signal is deasserted
- Data not ready until the wait signal is deasserted
Obtaining Access to the Bus: Bus Arbitration

Bus arbitration decides which device (bus master) gets the use of the bus next. Several schemes exist:

• A single bus master:
  – All bus requests are controlled by the processor.

• Daisy chain arbitration:
  – A bus grant line runs through the devices from the highest priority to lowest (priority determined by the position on the bus).
  – A high-priority device intercepts the bus grant signal, not allowing a low-priority device to see it (VME bus).

• Centralized, parallel arbitration:
  – Multiple request lines for each device.
  – A centralized arbiter chooses a requesting device and notifies it that it is now the bus master.
Obtaining Access to the Bus: Bus Arbitration

• Distributed arbitration by self-selection:
  – Use multiple request lines for each device
  – Each device requesting the bus places a code indicating its identity on the bus.
  – The requesting devices determine the highest priority device to control the bus.
  – Requires more lines for request signals (Apple NuBus).

• Distributed arbitration by collision detection:
  – Each device independently request the bus.
  – Multiple simultaneous requests result in a collision.
  – The collision is detected and a scheme to decide among the colliding requests is used (Ethernet).
Split-transaction Bus

- Used when multiple bus masters are present,
- Also known as a pipelined or a packet-switched bus
- The bus is available to other bus masters while a memory operation is in progress
  ⇒ Higher bus bandwidth, but also higher bus latency

A read transaction is tagged and broken into:
- A read request-transaction containing the address
- A memory-reply transaction that contains the data
  ⇒ address on the bus refers to a later memory access
Asynchronous Bus Operation

- Not clocked, instead self-timed using hand-shaking protocols between senders and receivers

A bus master performing a write:
- Master obtains control and asserts address, direction, and data
- Wait for a specified time for slaves to decode target
  - t1: Master asserts request line
  - t2: Slave asserts ack
  - t3: Master releases request
  - t4: Slave releases ack
## Examples of I/O Buses

<table>
<thead>
<tr>
<th>Bus</th>
<th>SBus</th>
<th>TurboChannel</th>
<th>MicroChannel</th>
<th>PCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originator</td>
<td>Sun</td>
<td>DEC</td>
<td>IBM</td>
<td>Intel</td>
</tr>
<tr>
<td>Clock Rate (MHz)</td>
<td>16-25</td>
<td>12.5-25</td>
<td>async</td>
<td>33</td>
</tr>
<tr>
<td>Addressing</td>
<td>Virtual</td>
<td>Physical</td>
<td>Physical</td>
<td>Physical</td>
</tr>
<tr>
<td>Data Sizes (bits)</td>
<td>8,16,32</td>
<td>8,16,24,32</td>
<td>8,16,24,32,64</td>
<td>8,16,24,32,64</td>
</tr>
<tr>
<td>Master</td>
<td>Multi</td>
<td>Single</td>
<td>Multi</td>
<td>Multi</td>
</tr>
<tr>
<td>Arbitration</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
</tr>
<tr>
<td>32 bit read (MB/s)</td>
<td>33</td>
<td>25</td>
<td>20</td>
<td>33</td>
</tr>
<tr>
<td>Peak (MB/s)</td>
<td>89</td>
<td>84</td>
<td>75</td>
<td>111 (222)</td>
</tr>
<tr>
<td>Max Power (W)</td>
<td>16</td>
<td>26</td>
<td>13</td>
<td>25</td>
</tr>
</tbody>
</table>
### Examples of CPU-Memory Buses

<table>
<thead>
<tr>
<th></th>
<th>Summit</th>
<th>Challenge</th>
<th>XDBus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Originator</strong></td>
<td>HP</td>
<td>SGI</td>
<td>Sun</td>
</tr>
<tr>
<td><strong>Clock Rate (MHz)</strong></td>
<td>60</td>
<td>48</td>
<td>66</td>
</tr>
<tr>
<td><strong>Split transaction?</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Address lines</strong></td>
<td>48</td>
<td>40</td>
<td>??</td>
</tr>
<tr>
<td><strong>Data lines</strong></td>
<td>128</td>
<td>256</td>
<td>144 (parity)</td>
</tr>
<tr>
<td><strong>Data Sizes (bits)</strong></td>
<td>512</td>
<td>1024</td>
<td>512</td>
</tr>
<tr>
<td><strong>Clocks/transfer</strong></td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td><strong>Peak (MB/s)</strong></td>
<td>960</td>
<td>1200</td>
<td>1056</td>
</tr>
<tr>
<td><strong>Master</strong></td>
<td>Multi</td>
<td>Multi</td>
<td>Multi</td>
</tr>
<tr>
<td><strong>Arbitration</strong></td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
</tr>
<tr>
<td><strong>Addressing</strong></td>
<td>Physical</td>
<td>Physical</td>
<td>Physical</td>
</tr>
<tr>
<td><strong>Slots</strong></td>
<td>16</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td><strong>Busses/system</strong></td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>13 inches</td>
<td>12 inches</td>
<td>17 inches</td>
</tr>
</tbody>
</table>
SCSI: Small Computer System Interface

- Clock rate: 5 MHz / 10 MHz (fast) / 20 MHz (ultra).
- Width: $n = 8$ bits / 16 bits (wide); up to $n - 1$ devices to communicate on a bus or “string”.
- Devices can be slave (“target”) or master (“initiator”).
- SCSI protocol: A series of “phases”, during which specific actions are taken by the controller and the SCSI disks and devices.
  - Bus Free: No device is currently accessing the bus
  - Arbitration: When the SCSI bus goes free, multiple devices may request (arbitrate for) the bus; fixed priority by address
  - Selection: Informs the target that it will participate (Reselection if disconnected)
  - Command: The initiator reads the SCSI command bytes from host memory and sends them to the target
  - Data Transfer: data in or out, initiator: target
  - Message Phase: message in or out, initiator: target (identify, save/restore data pointer, disconnect, command complete)
  - Status Phase: target, just before command complete
SCSI “Bus”: Channel Architecture

Peer-to-peer protocols
Initiator/target
Linear byte streams
Disconnect/reconnect

Command Setup
- Arbitration
- Selection
- Message Out (Identify)
- Command

Data Transfer
- Data In

Command Completion
- Status
- Message In (Command Complete)

Disconnect to seek/½ ll buffer
Message In (Disconnect)
- - Bus Free - -
  Arbitration
  Reselection
Message In (Identify)

If no disconnect is needed

Disconnect to ¼ ll buffer
Message In (Save Data Ptr)
Message In (Disconnect)
- - Bus Free - -
  Arbitration
  Reselection
Message In (Identify)
Message In (Restore Data Ptr)

Message In (Identify)
- - Bus Free - -
  Arbitration
  Reselection

Message In (Disconnect)
I/O Data Transfer Methods

- **Programmed I/O (PIO):** Polling
  - The I/O device puts its status information in a status register.
  - The processor must periodically check the status register.
  - The processor is totally in control and does all the work.
  - Very wasteful of processor time.

- **Interrupt-Driven I/O:**
  - An interrupt line from the I/O device to the CPU is used to generate an I/O interrupt indicating that the I/O device needs CPU attention.
  - The interrupting device places its identity in an interrupt vector.
  - Once an I/O interrupt is detected the current instruction is completed and an I/O interrupt handling routine is executed to service the device.
I/O data transfer methods

Direct Memory Access (DMA):

- Implemented with a specialized controller that transfers data between an I/O device and memory independent of the processor.
- The DMA controller becomes the bus master and directs reads and writes between itself and memory.
- Interrupts are still used only on completion of the transfer or when an error occurs.
- DMA transfer steps:
  - The CPU sets up DMA by supplying device identity, operation, memory address of source and destination of data, the number of bytes to be transferred.
  - The DMA controller starts the operation. When the data is available it transfers the data, including generating memory addresses for data to be transferred.
  - Once the DMA transfer is complete, the controller interrupts the processor, which determines whether the entire operation is complete.
DMA and Virtual Memory: Virtual DMA

Allow the DMA mechanism to use virtual addresses that are mapped directly to physical addresses using the DMA:

- A buffer is sequential in virtual memory but the pages can be scattered in physical memory.
- Pages may need to be “locked” by the operating system if a process is moved.
Cache & I/O: The Stale Data Problem

- Three copies of data, may exist in: cache, memory, disk.
  ⇒ Similar to cache coherency problem in multiprocessor systems.

- CPU or I/O may modify one copy while other copies contain stale data.

- Possible solutions:
  - Connect I/O directly to CPU cache; CPU performance suffers.
  - With write-back cache, the operating system flushes output addresses to make sure data is not in cache.
  - Use write-through cache; I/O receives updated data from memory.
  - The operating system designates memory addresses involved in input operations as non-cacheable.
I/O Connected Directly To Cache

CPU-memory bus

- Cache
- TLB
- CPU
- Bus adapter

Main memory

I/O bus

- I/O controller
  - Disk
- I/O controller
  - Disk
- I/O controller
  - Graphics output
- I/O controller
  - Network
I/O Performance Metrics

- Diversity: The variety of I/O devices that can be connected to the system.
- Capacity: The maximum number of I/O devices that can be connected to the system.
- Producer/server Model of I/O: The producer (CPU, human etc.) creates tasks to be performed and places them in a task buffer (queue); the server (I/O device or controller) takes tasks from the queue and performs them.
- I/O Throughput: The maximum data rate that can be transferred to/from an I/O device or sub-system, or the maximum number of I/O tasks or transactions completed by I/O in a certain period of time
  ⇒ Maximized when task buffer is never empty.
- I/O Latency or response time: The time an I/O task takes from the time it is placed in the task buffer or queue until the server (I/O system) finishes the task. Includes buffer waiting or queuing time.
  ⇒ Maximized when task buffer is always empty.
I/O Performance Metrics: Throughput:

- Throughput is a measure of speed—the rate at which the storage system delivers data.

- Throughput is measured in two ways:
  - I/O rate, measured in accesses/second:
    - I/O rate is generally used for applications where the size of each request is small, such as transaction processing.
  - Data rate, measured in bytes/second or megabytes/second (MB/s).
    - Data rate is generally used for applications where the size of each request is large, such as scientific applications.
I/O Performance Metrics: Response time

• Response time measures how long a storage system takes to access data. This time can be measured in several ways. For example:
  – One could measure time from the user’s perspective,
  – the operating system’s perspective,
  – or the disk controller’s perspective, depending on what you view as the storage system.
Producer-Server Model

Response Time = Time_{\text{System}} = Time_{\text{Queue}} + Time_{\text{Server}}

Throughput vs. Response Time

Response time (latency) in ms

Percent of maximum throughput (bandwidth)
Components of A User/Computer System Transaction

• In an interactive user/computer environment, each interaction or transaction has three parts:

  – Entry Time: Time for user to enter a command

  – System Response Time: Time between user entry & system reply.

  – Think Time: Time from response until user begins next command.
User/Interactive Computer Transaction Time

Workload
- Conventional interactive workload (1.0 sec. system response time)
- Conventional interactive workload (0.3 sec. system response time)
- High-function graphics workload (1.0 sec. system response time)
- High-function graphics workload (0.3 sec. system response time)

Time (seconds)

- Entry time
- System response time
- Think time

-34% total (-70% think)
-70% total (-81% think)
Introduction to Queuing Theory

- Concerned with long term, steady state than in startup:
  - where \( \Rightarrow \) Arrivals = Departures

- **Little’s Law:**
  
  Mean number tasks in system = arrival rate \( \times \) mean response time

- Applies to any system in equilibrium, as long as nothing in the black box is creating or destroying tasks.
I/O Performance & Little’s Queuing Law

Given: An I/O system in equilibrium (input rate is equal to output rate) and:

- $T_{ser}$: Average time to service a task
- $T_q$: Average time per task in the queue
- $T_{sys}$: Average time per task in the system, or the response time, the sum of $T_{ser}$ and $T_q$
- $r$: Average number of arriving tasks/sec
- $L_{ser}$: Average number of tasks in service.
- $L_q$: Average length of queue
- $L_{sys}$: Average number of tasks in the system, the sum of $L_q$ and $L_{ser}$

Little’s Law states: $L_{sys} = r \times T_{sys}$

Server utilization = $u = r / \text{Service rate} = r \times T_{ser}$

$u$ must be between 0 and 1 otherwise there would be more tasks arriving than could be serviced.
• Service time completions vs. waiting time for a busy server: randomly arriving event joins a queue of arbitrary length when server is busy, otherwise serviced immediately
  – Unlimited length queues key simplification

• A single server queue: combination of a servicing facility that accommodates 1 customer at a time (server) + waiting area (queue): together called a system

• Server spends a variable amount of time with customers; how do you characterize variability?
  – Distribution of a random variable: histogram? curve?
A Little Queuing Theory

System

- Server spends a variable amount of time with customers
  - Weighted mean time \( m_1 = (f_1 \times T_1 + f_2 \times T_2 + ... + f_n \times T_n)/F \)
    - where \( (F=f_1 + f_2...) \)
  - Variance = \( (f_1 \times T_1^2 + f_2 \times T_2^2 + ... + f_n \times T_n^2)/F - m_1^2 \)
    - Must keep track of unit of measure (100 ms\(^2\) vs. 0.1 s\(^2\))
  - Squared coefficient of variance: \( C = \text{variance}/m_1^2 \)
    - Unitless measure (100 ms\(^2\) vs. 0.1 s\(^2\))

- Exponential distribution \( C = 1 \): most short relative to average, few others long;
  90% < 2.3 x average, 63% < average

- Hypoexponential distribution \( C < 1 \): most close to average,
  \( C=0.5 \Rightarrow 90\% < 2.0 \times \text{average}, \text{only} 57\% < \text{average} \)

- Hyperexponential distribution \( C > 1 \): further from average
  \( C=2.0 \Rightarrow 90\% < 2.8 \times \text{average}, 69\% < \text{average} \)
A Little Queuing Theory: Average Wait Time

- Calculating average wait time in queue $T_q$
  - If something at server, it takes to complete on average $m_1(z)$
  - Chance server is busy = $u$; average delay is $u \times m_1(z)$
  - All customers in line must complete; each avg $T_{ser}$
    
    
    $T_q = u \times m_1(z) + L_q \times T_{ser} = \frac{1}{2} \times u \times T_{ser} \times (1 + C) + \frac{L_q \times T_{ser}}{1/2 \times u \times T_{ser}}$
    
    $T_q = \frac{1}{2} \times u \times T_{ser} \times (1 + C) + \frac{T_q \times T_{ser}}{u \times T_q}$
    
    $T_q \times (1 - u) = T_{ser} \times u \times (1 + C) / 2$
    
    $T_q = T_{ser} \times u \times (1 + C) / (2 \times (1 - u))$

- Notation:
  
  $r$ average number of arriving customers/second
  $T_{ser}$ average time to service a customer
  $u$ server utilization (0..1): $u = r \times T_{ser}$
  $T_q$ average time/customer in queue
  $L_q$ average length of queue: $L_q = r \times T_q$
A Little Queuing Theory: M/G/1 and M/M/1

- Assumptions so far:
  - System in equilibrium
  - Time between two successive arrivals in line are random
  - Server can start on next customer immediately after prior finishes
  - No limit to the queue: works First-In-First-Out
  - Afterward, all customers in line must complete; each avg $T_{ser}$

- Described “memoryless” or Markovian request arrival (M for C=1 exponentially random), General service distribution (no restrictions), 1 server: M/G/1 queue

- When Service times have C = 1, M/M/1 queue

$$T_q = T_{ser} \times u \times \frac{1 + C}{2 \times (1 - u)} = T_{ser} \times u / (1 - u)$$

$T_{ser}$ average time to service a customer

$u$ server utilization (0..1): $u = r \times T_{ser}$

$T_q$ average time/customer in queue
I/O Queuing Performance: An Example

• A processor sends 10 x 8KB disk I/O requests per second, requests & service are exponentially distributed, average disk service time = 20 ms

• On average:
  – How utilized is the disk, $u$?
  – What is the average time spent in the queue, $T_q$?
  – What is the average response time for a disk request, $T_{sys}$?
  – What is the number of requests in the queue $L_q$? In system, $L_{sys}$?

• We have:

  \[
  r \quad \text{average number of arriving requests/second} = 10
  \]

  \[
  T_{ser} \quad \text{average time to service a request} = 20 \text{ ms (0.02s)}
  \]

• We obtain:

  \[
  u \quad \text{server utilization: } u = r \times T_{ser} = 10/s \times 0.02s = 0.2
  \]

  \[
  T_q \quad \text{average time/request in queue} = T_{ser} \times u \times (1 - u)
  \]

  \[
  = 20 \times 0.2/(1-0.2) = 20 \times 0.25 = 5 \text{ ms (0.005s)}
  \]

  \[
  T_{sys} \quad \text{average time/request in system: } T_{sys} = T_q + T_{ser} = 25 \text{ ms}
  \]

  \[
  L_q \quad \text{average length of queue: } L_q = r \times T_q
  \]

  \[
  = 10/s \times 0.005s = 0.05 \text{ requests in queue}
  \]

  \[
  L_{sys} \quad \text{average # tasks in system: } L_{sys} = r \times T_{sys} = 10/s \times 0.025s = 0.25
  \]
I/O Queuing Performance: An Example

- Previous example with a faster disk with average disk service time = 10 ms
- The processor still sends 10 x 8KB disk I/O requests per second, requests & service are exponentially distributed

- On average:
  - How utilized is the disk, $u$?
  - What is the average time spent in the queue, $T_q$?
  - What is the average response time for a disk request, $T_{sys}$?

- We have:
  - $r$ average number of arriving requests/second = 10
  - $T_{ser}$ average time to service a request = 10 ms (0.01s)

- We obtain:
  - $u$ server utilization: $u = r \times T_{ser} = 10/s \times .01s = 0.1$
  - $T_q$ average time/request in queue $= T_{ser} \times u / (1-u)$
    $= 10 \times 0.1/(1-0.1) = 10 \times 0.11 = 1.11$ ms (0.0011s)
  - $T_{sys}$ average time/request in system: $T_{sys} = T_q + T_{ser} = 10 + 1.11 = 11.11$ ms

response time is $25/11.11 = 2.25$ times faster even though the new service time is only 2 times faster.
Designing an I/O System

• When designing an I/O system, the components that make it up should be balanced.

• Six steps for designing an I/O systems are
  – List types of devices and buses in system
  – List physical requirements (e.g., volume, power, connectors, etc.)
  – List cost of each device, including controller if needed
  – Record the CPU resource demands of device
    • CPU clock cycles directly for I/O (e.g. initiate, interrupts, complete)
    • CPU clock cycles due to stalls waiting for I/O
    • CPU clock cycles to recover from I/O activity (e.g., cache flush)
  – List memory and I/O bus resource demands
  – Assess the performance of the different ways to organize these devices
Example: Reading a Page from Disk Directly into Cache

• What is the impact on the CPU performance of reading a disk page directly into the cache?

• Assumptions:
  – Each page is 16 KB and cache block size is 64 bytes
  – One page is brought in every 1 million clock cycles
  – Addresses of new page are not in cache.
  – CPU does not access data in the new page
  – 95% of blocks displaced from cache will later cause misses
  – The cache uses write back, with an average of 50% dirty blocks
  – The are 15,000 cache misses every 1 million clock cycles if no I/O
  – The miss penalty is 30 clock cycles (plus 30 to write back if dirty)
Example: Reading a Page from Disk Directly into Cache

- The number of extra cycles due to I/O is computed as
  - \((16384 \text{ bytes/page})/(64 \text{ bytes/block}) = 256 \text{ blocks/page}\)
  - Cycles due to writing 50\% of blocks back to memory
    - \(50\% \times 256 \times 30 = 3,840 \text{ cycles}\)
  - Cycles due to replacing on 95\% of blocks - all dirty
    - \(95\% \times 256 \times (30 + 30) = 14,952 \text{ cycles}\)
  - Total extra cycles from I/O = 3,840 + 14,952 = 18,432

- Number of cycles if no I/O is
  - \(1,000,000 + 15,000 + 50\% \times 15,000 = 1,675,000\)

- Overhead due to I/O is
  - \(18,432/1,675,000 = 1.1\%\)
Example: Determining the I/O Bottleneck

• Assume the following system components:
  – 500 MIPS CPU
  – 16-byte wide memory system with 100 ns cycle time
  – 200 MB/sec I/O bus
  – 20 20 MB/sec SCSI-2 buses, with 1 ms controller overhead
  – 5 disks per SCSI bus: 8 ms seek, 7,200 RPMS, 6MB/sec

• Other assumptions
  – All devices used to 100% capacity, always have average values
  – Average I/O size is 16 KB
  – OS uses 10,000 CPU instr. for a disk I/O

• What is the average IOPS? What is the average bandwidth?
Example: Determining the I/O Bottleneck

• The performance of I/O systems is determined by the portion with the lowest I/O bandwidth
  – CPU: \( \frac{500 \text{ MIPS}}{10,000 \text{ instr. per I/O}} = 50,000 \text{ IOPS} \)
  – Main Memory: \( \frac{16 \text{ bytes}}{100 \text{ ns} \times 16 \text{ KB per I/O}} = 10,000 \text{ IOPS} \)
  – I/O bus: \( \frac{200 \text{ MB/sec}}{16 \text{ KB per I/O}} = 12,500 \text{ IOPS} \)
  – SCSI-2: \( \frac{20 \text{ buses}}{(1 \text{ ms} + \frac{16 \text{ KB}}{20 \text{ MB/sec}}) \text{ per I/O}} = 11,120 \text{ IOPS} \)
  – Disks: \( \frac{100 \text{ disks}}{(8 \text{ ms} + 0.5/(7200 \text{ RPMS}) + \frac{16 \text{ KB}}{6 \text{ MB/sec}}) \text{ per I/O}} = 6,700 \text{ IOPS} \)

• In this case, the disks limit the I/O performance to 6,700 IOPS

• The average I/O bandwidth is
  – 6,700 IOPS \times (16 \text{ KB/sec}) = 107.2 \text{ MB/sec}