Input/Output & System Performance Issues

- System I/O Connection Structure
  - Types of Buses in the system.
- I/O Data Transfer Methods.
- Cache & I/O.
- I/O Performance Metrics.
- Magnetic Disk Characteristics.
- I/O System Modeling Using Queuing Theory.
- Designing an I/O System & System Performance:
  - System performance bottleneck.

In textbook: Ch. 7.1-7.3, 7.7, 7.8
System Components

CPU Core
1 GHz - 3.0 GHz
4-way Superscaler
RISC or RISC-core (x86):
  Deep Instruction Pipelines
  Dynamic scheduling
  Multiple FP, integer FUs
  Dynamic branch prediction
  Hardware speculation

SDRAM
PC100/PC133
100-133MHz
64-128 bits wide
2-way interleaved
~ 900 MBYTES/SEC (64bit)

Double Data Rate (DDR) SDRAM
PC3200
200 MHz DDR
64-128 bits wide
4-way interleaved
~3.3 GBYTES/SEC (64bit)

RAMbus DRAM (RDRAM)
400MHz DDR
16 bits wide (32 banks)
~ 1.6 GBYTES/SEC

CPU

Caches

L1 16-128K 1-2 way set associative (on chip), separate or unified
L2 256K-2M 4-32 way set associative (on chip) unified
L3 2-16M 8-32 way set associative (on or off chip) unified

Memory

Memory Controller

Bus Adapter

I/O Controllers

Disks
Displays
Keyboards

Networks

I/O Devices

Main I/O Bus
Example: PCI, 33-66MHz
32-64 bits wide
133-528 MB/s
PCI-X 133MHz 64-bits wide
1066 MB/s

I/O Subsystem

FSB
Front System Bus

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

Example:
Alpha, AMD K7: EV6, 200-400 MHz
Intel PII, PIII: GTL+ 133 MHz
Intel P4 800 MHz
Types of Buses in The System

Processor-Memory Bus (Front System Bus, FSB):

- Offers very high-speed and low latency.
- Matched to the memory system to maximize memory-processor bandwidth.
- Usually design-specific.
- Examples: Alpha EV6 (AMD K7), Peak bandwidth = 266 Mhz x 8 = 2.1 GB/s
  Intel GTL+ (P3), Peak bandwidth = 133 Mhz x 8 = 1 GB/s
  Intel P4, Peak bandwidth = 800 Mhz x 8 = 6.4 GB/s

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 instead connected processor-memory bus via a bus adapter (chipset south bridge).
- Examples: PCI, PCI-X, SCSI.
## 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., 64 bits)</td>
<td>Narrower is cheaper (e.g., 16 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>
I/O Interface

I/O Interface, I/O 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 (usually memory-mapped) 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
  • (e.g IDE, SCSI)

– Handles I/O details (originally done by CPU) 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.
## Main System IO Bus Example: PCI

<table>
<thead>
<tr>
<th>Specification</th>
<th>Bus Width (bits)</th>
<th>Bus Frequency (MHz)</th>
<th>Peak Bandwidth (MB/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI 2.2</td>
<td>32</td>
<td>33.3</td>
<td>133</td>
</tr>
<tr>
<td>PCI 2.2</td>
<td>64</td>
<td>33.3</td>
<td>266</td>
</tr>
<tr>
<td>PCI 2.2</td>
<td>64</td>
<td>66.6</td>
<td>533</td>
</tr>
<tr>
<td>PCI-X 1.0</td>
<td>64</td>
<td>133.3</td>
<td>1066</td>
</tr>
</tbody>
</table>

### Addressing
- **Master**: Physical
- **Arbitration**: Multi
- **Central**

### PCI Bus Transaction Latency:
- PCI requires 9 cycles @ 33Mhz (272ns)
- PCI-X requires 10 cycles @ 133MHz (75ns)
# Storage IO Interfaces/Buses

<table>
<thead>
<tr>
<th></th>
<th>IDE/Ultra ATA</th>
<th>SCSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Width</td>
<td>16 bits</td>
<td>8 or 16 bits (wide)</td>
</tr>
<tr>
<td>Clock Rate</td>
<td>Upto 100MHz</td>
<td>10MHz (Fast)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20MHz (Ultra)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40MHz (Ultra2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80MHz (Ultra3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>160MHz (Ultra4)</td>
</tr>
<tr>
<td>Bus Masters</td>
<td>1</td>
<td>Multiple</td>
</tr>
<tr>
<td>Max no. devices</td>
<td>2</td>
<td>7 (8-bit bus)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 (16-bit bus)</td>
</tr>
<tr>
<td>Peak Bandwidth</td>
<td>200 MB/s</td>
<td>320MB/s (Ultra4)</td>
</tr>
</tbody>
</table>
## Example CPU-Memory Buses
### (Front System Buses, FSBs)

<table>
<thead>
<tr>
<th>Bus</th>
<th>Summit</th>
<th>Challenge</th>
<th>XDBus</th>
<th>SP</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originator</td>
<td>HP</td>
<td>SGI</td>
<td>Sun</td>
<td>IBM</td>
<td>Intel</td>
</tr>
<tr>
<td>Clock Rate (MHz)</td>
<td>60</td>
<td>48</td>
<td>66</td>
<td>111</td>
<td>800</td>
</tr>
<tr>
<td>Split transaction?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Address lines</td>
<td>48</td>
<td>40</td>
<td>??</td>
<td>??</td>
<td>??</td>
</tr>
<tr>
<td>Data lines</td>
<td>128</td>
<td>256</td>
<td>144</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Clocks/transfer</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>??</td>
<td>??</td>
</tr>
<tr>
<td>Peak (MB/s)</td>
<td>960</td>
<td>1200</td>
<td>1056</td>
<td>1700</td>
<td>6400</td>
</tr>
<tr>
<td>Master</td>
<td>Multi</td>
<td>Multi</td>
<td>Multi</td>
<td>Multi</td>
<td>Multi</td>
</tr>
<tr>
<td>Arbitration</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
<td>Central</td>
</tr>
<tr>
<td>Addressing</td>
<td>Physical</td>
<td>Physical</td>
<td>Physical</td>
<td>Physical</td>
<td>Physical</td>
</tr>
<tr>
<td>Length</td>
<td>13 inches</td>
<td>12 inches</td>
<td>17 inches</td>
<td>??</td>
<td>??</td>
</tr>
</tbody>
</table>
I/O Data Transfer Methods

• **Programmed I/O (PIO):** Polling (For low-speed I/O)
  – 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.
  – Used for low-speed I/O devices (mice, keyboards etc.)

• **Interrupt-Driven I/O (For medium-speed 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 (by OS) is executed to service the device.
I/O data transfer methods

Direct Memory Access (DMA) (For high-speed I/O):

- 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.
- Low CPU overhead, used in high speed I/O (storage, network interfaces)
- 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.
I/O Controller Architecture

Peripheral or Main I/O Bus (PCI, PCI-X, etc.)

- Host Memory
- Processor Cache
- Host Processor

Peripheral Bus Interface/DMA

- Micro-controller or Embedded processor
- $\mu$Proc

Buffer Memory

ROM

I/O Channel Interface

I/O Controller

Chipset
- North Bridge
- South Bridge
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 (old) 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 (used too much memory bandwidth).
  - The operating system designates memory addresses involved in I/O operations as non-cacheable.
I/O Connected Directly To Cache

This solution may slow down CPU performance.
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
    \[ \Rightarrow 5 \text{ times system performance increase} \]
    \[ \Rightarrow (50\% \text{ loss in performance}) \]
  - Increasing CPU performance by 100 times
    \[ \Rightarrow \sim 10 \text{ times system performance} \]
    \[ \Rightarrow (90\% \text{ loss of performance}) \]

- The I/O system performance bottleneck diminishes the benefit of faster CPUs on overall system performance.
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.
Factors Affecting System I/O Processing Performance

- **I/O processing computational requirements:**
  - CPU computations available for I/O operations.
  - Operating system I/O processing policies/routines.
  - I/O Data Transfer Method used.
    - CPU cycles needed: Polling >> Interrupt Driven > DMA

- **I/O Subsystem performance:**
  - Raw performance of I/O devices (i.e. magnetic disk performance).
  - I/O bus capabilities.
  - I/O subsystem organization.
  - Loading level of I/O devices (queuing delay, response time).

- **Memory subsystem performance:**
  - Available memory bandwidth for I/O operations.
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,
      - Transactions Per Second (TPS) or,
      - I/O Operations Per Second (IOPS).
    - I/O rate is generally used for applications where the size of each request is small, such as in 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 in scientific applications.
I/O Performance Metrics: Response time

• Response time measures how long a storage (or I/O) 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 or I/O system.
Producer-Server Model

User or CPU

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

Throughput vs. Response Time

Queue almost empty most of the time
Less time in queue

Queue full most of the time
More time in queue
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

<table>
<thead>
<tr>
<th>Workload</th>
<th>Entry time</th>
<th>System response time</th>
<th>Think time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional interactive workload</td>
<td>0</td>
<td>70%</td>
<td>30%</td>
</tr>
<tr>
<td>(1.0 sec. system response time)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conventional interactive workload</td>
<td>0</td>
<td>70%</td>
<td>30%</td>
</tr>
<tr>
<td>(0.3 sec. system response time)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-function graphics workload</td>
<td>0</td>
<td>10%</td>
<td>90%</td>
</tr>
<tr>
<td>(1.0 sec. system response time)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-function graphics workload</td>
<td>0</td>
<td>10%</td>
<td>90%</td>
</tr>
<tr>
<td>(0.3 sec. system response time)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Magnetic Disks

Characteristics:

- Diameter: 2.5in - 5.25in
- Rotational speed: 3,600 RPM - 15,000 RPM
- Tracks per surface.
- Sectors per track: Outer tracks contain more sectors.
- Recording or Areal Density: Tracks/in X Bits/in
- Cost Per Megabyte.
- Seek Time: (2-12 ms)
  The time needed to move the read/write head arm.
  Reported values: Minimum, Maximum, Average.
- Rotation Latency or Delay: (2-8 ms)
  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 =

  average seek time + average rotational delay + transfer time
  + disk controller overhead

  (ignoring queuing time)
Disk Performance Example

- Given the following Disk Parameters:
  - Average seek time is 5 ms
  - Disk spins at 10,000 RPM
  - Transfer rate is 40 MB/sec

- Controller overhead is 0.1 ms

- Assume that the disk is idle, so no queuing delay exist.

- What is Average Disk read or write time for a 512-byte Sector?

  Avg. seek + ave. rot delay + transfer time + controller overhead

  \[ 5 \text{ ms} + \frac{0.5}{(10000 \text{ RPM}/60)} + 0.5 \text{ KB}/40 \text{ MB/s} + 0.1 \text{ ms} \]

  \[ 5 + 3 + 0.13 + 0.1 = 8.23 \text{ ms} \]

  Disk Service Time for this request
Introduction to Queuing Theory

• Concerned with long term, steady state than in startup:
  – where => Arrivals = Departures
  – Rate = Rate

• **Little’s Law:**
  Mean number tasks in system = arrival rate x 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 = 1/Service rate
  - $T_{q}$: 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.
A Little Queuing Theory

- 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

- Server spends a variable amount of time with customers
  - Weighted mean time $m_1 = (f_1 \times T_1 + f_2 \times T_2 + \ldots + f_n \times T_n)/F$
    - where $(F=f_1 + f_2 \ldots)$
  - Variance $= (f_1 \times T_1^2 + f_2 \times T_2^2 + \ldots + f_n \times T_n^2)/F - m_1^2$
    - 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 x average, only 57% < average

- Hyperexponential distribution $C > 1$: further from average, $C=2.0 \Rightarrow$ 90% < 2.8 x average, 69% < 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 $m1(z)$
  – Chance server is busy = $u$; average delay is $u \times m1(z)$
  – All customers in line must complete; each avg $T_{ser}$

$$T_q = u \times m1(z) + L_q \times T_{ser} = \frac{1}{2} \times u \times T_{ser} \times (1 + C) + L_q \times T_{ser}$$

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

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

• 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*

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

- $T_{ser}$ average time to service a customer
- $u$ server utilization (0..1): $u = r \times T_{ser}$
- $T_q$ average time/customer in queue
- $L_q$ Average length of queue $L_q = r \times T_q$
Multiple Server (Disk/Controller) I/O Modeling:

- **M/M/m Queue**

- I/O system with Markovian request arrival rate \( r \)
- A single queue serviced by \( m \) servers (disks + controllers) each with Markovian Service rate = \( 1/T_{ser} \)

\[
T_q = T_{ser} \times u \quad /\,[m \,(1-u)]
\]

\[
u = r \times T_{ser} / m
\]

- \( m \) number of servers
- \( T_{ser} \) average time to service a customer
- \( u \) server utilization (0..1): \( u = r \times T_{ser} / m \)
- \( T_q \) average time/customer in queue
I/O Queuing Performance: An M/M/1 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:
  - \( r \) average number of arriving requests/second = 10
  - \( T_{ser} \) average time to service a request = 20 ms (0.02s)

- We obtain:
  - \( u \) server utilization: \( u = r \times T_{ser} = 10/s \times .02s = 0.2 = 20\% \)
  - \( T_q \) average time/request in queue: \( T_q = T_{ser} \times u / (1 - u) \)
    \( = 20 \times 0.2/(1-0.2) = 20 \times 0.25 = 5 \text{ ms (0.005s)} \)
  - \( T_{sys} \) average time/request in system: \( T_{sys} = T_q + T_{ser} = 25 \text{ ms} \)
  - \( L_q \) average length of queue: \( L_q = r \times T_q \)
    \( = 10/s \times .005s = 0.05 \text{ requests in queue} \)
  - \( L_{sys} \) average # tasks in system: \( L_{sys} = r \times T_{sys} = 10/s \times .025s = 0.25 \)
I/O Queuing Performance: An M/M/1 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: $u = r \times T_{ser} = 10/s \times .01s = 0.1 = 10\%$
- $T_q$ average time/request in queue: $T_q = T_{ser} \times u / (1 - u) = 10 \times .1/(1-.1) = 10 \times 0.11 = 1.11$ 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 due to lower queuing time.
Designing an I/O System

• When designing an I/O system, the **performance** of 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 system performance of the different ways to organize these devices:
    • For each system configuration identify which system component (CPU, memory, I/O buses, I/O devices etc.) is the **performance bottleneck**.
Example: Determining the System I/O Bottleneck (ignoring queuing delays)

• 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 instructions for a disk I/O
  – Ignore disk/controller queuing delays.

• What is the average IOPS? What is the average I/O bandwidth?
Example: Determining the I/O Bottleneck (ignoring queuing delays)

- 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
  - \(6,700 \text{ IOPS} \times (16 \text{ KB/sec}) = 107.2 \text{ MB/sec}\)
Example: Determining the I/O Bottleneck
Accounting For I/O Queue Time (M/M/m queue)

• 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 60% capacity (i.e maximum utilization allowed).
  – Treat the I/O system as an M/M/m queue.
  – Requests are assumed spread evenly on all disks.
  – Average I/O size is 16 KB
  – OS uses 10,000 CPU instructions for a disk I/O

• What is the average IOPS? What is the average bandwidth?
• Average response time per IO operation?
Example: Determining the I/O Bottleneck

Accounting For I/O Queue Time (M/M/m queue)

- The performance of I/O systems is still determined by the system component with the lowest I/O bandwidth
  - CPU: \(\frac{500 \text{ MIPS}}{10,000 \text{ instr. per I/O}} \times 0.6 = 30,000 \text{ IOPS}\)
    - CPU time per I/O = \(\frac{10,000}{500,000,000} = 0.02 \text{ ms}\)
  - Main Memory: \(\frac{16 \text{ bytes}}{100 \text{ ns} \times 16 \text{ KB per I/O}} \times 0.6 = 6,000 \text{ IOPS}\)
    - Memory time per I/O = \(\frac{1}{10,000} = 0.1 \text{ ms}\)
  - I/O bus: \(\frac{200 \text{ MB/sec}}{16 \text{ KB per I/O}} \times 0.6 = 12,500 \text{ IOPS}\)
  - SCSI-2: \(\frac{20 \text{ buses}}{(1 \text{ ms} + \frac{16 \text{ KB}}{20 \text{ MB/sec}}) \text{ per I/O}} = 7,500 \text{ IOPS}\)
    - SCSI bus time per I/O = \(1\text{ms} + \frac{16}{20} \text{ms} = 1.8\text{ms}\)
  - Disks: \(\frac{100 \text{ disks}}{(8 \text{ ms} + \frac{0.5}{7200 \text{ RPMs}} + \frac{16 \text{ KB}}{6 \text{ MB/sec}}) \text{ per I/O}} \times 0.6 = 6,700 \times 0.6 = 4020 \text{ IOPS}\)
    - \(T_{\text{ser}} = 8 \text{ ms} + 0.5/(7200 \text{ RPMs}) + (16 \text{ KB})/(6 \text{ MB/sec}) = 8 + 4.2 + 2.7 = 14.9 \text{ ms}\)

- The disks limit the I/O performance to \(r = 4020 \text{ IOPS}\)
- The average I/O bandwidth is \(4020 \text{ IOPS} \times (16 \text{ KB/sec}) = 64.3 \text{ MB/sec}\)
- \(T_q = T_{\text{ser}} \times u / [m \ (1 – u)] = 14.9 \text{ ms} \times 0.6 / [100 \times 0.4] = 0.22 \text{ ms}\)
- \(\text{Response Time} = T_{\text{ser}} + T_q + T_{\text{cpu}} + T_{\text{memory}} + T_{\text{scsi}} = 14.9 + 0.22 + 0.02 + 0.1 + 1.8 = 17.04 \text{ ms}\)