Input/Output & System Performance Issues

• System Architecture & I/O Connection Structure
  – Types of Buses/Interconnects in the system.

• I/O Data Transfer Methods.

• System and I/O Performance Metrics.
  – I/O Throughput  i.e system throughput in tasks per second
  – I/O Latency (Response Time)  i.e Time it takes the system to process an average task

• Magnetic Disk Characteristics.

• I/O System Modeling Using Queuing Theory.
  – Little’s Queuing Law
  – Single Server/Single Queue I/O Modeling: M/M/1 Queue
  – Multiple Servers/Single Queue I/O Modeling: M/M/m Queue

• Designing an I/O System & System Performance:
  – Determining system performance bottleneck.
    • (i.e. which component creates a system performance bottleneck)

4th Edition: Chapter 6.1, 6.2, 6.4, 6.5
3rd Edition: Chapter 7.1-7.3, 7.7, 7.8
The Von-Neumann Computer Model

- Partitioning of the computing engine into components:
  1. **Central Processing Unit (CPU):** Control Unit (instruction decode, sequencing of operations), Datapath (registers, arithmetic and logic unit, buses).
  2. **Memory:** Instruction (program) and operand (data) storage.
  3. **Input/Output (I/O):** Communication between the CPU/memory and the outside world.

System Architecture = System components and how the components are connected (system interconnects)

**System performance depends on many aspects of the system**
("limited by weakest link in the chain"): The system performance bottleneck
Input and Output (I/O) Subsystem

- The I/O subsystem provides the mechanism for communication between the CPU and the outside world (I/O devices).

- Design factors:
  - I/O device characteristics (input, output, storage, etc.) /Performance.
  - I/O Connection Structure (degree of separation from memory operations).
  - I/O interface (the utilization of dedicated I/O and bus controllers).
  - Types of buses/system interconnects (processor-memory vs. I/O buses/interconnects).
  - I/O data transfer or synchronization method (programmed I/O, interrupt-driven, DMA).

Components of Total System Execution Time:
(or response time)

| CPU | Memory | I/O |

EECC551 - Shaaban

#3  Lec # 9  Fall 2009  10-27-2009
Current System Architecture:
Isolated I/O: Separate memory (system) and I/O buses.

System Architecture = System Components + System Component Interconnects

System Bus or Front Side Bus (FSB)

Typical System Architecture

Microprocessor Chip

Cache (One or more levels)

CPU

Back Side Bus (BSB)

I/O Controller Hub (Chipset South Bridge) i.e. System Core Logic

Memory Controller (Chipset North Bridge)

Main memory

I/O Subsystem

Bus adapter

Two Types of System Interconnects/Buses:
1- CPU-Memory Bus or interconnect
2 – I/O Buses/interfaces

Thus

EECC551 - Shaaban

#4  Lec # 9  Fall 2009  10-27-2009
Typical Mainstream System Architecture

System Architecture = System Components + System Component Interconnects

- **CPU Core**
  - 1 GHz - 3.8 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
  - ~3.2 GBYTES/SEC (64bit)

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

- **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

- **System Bus**

- **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**
  - Example: Alpha, AMD K7: EV6, 200-400 MHz
  - Intel PII, PIII: GTL+ 133 MHz
  - Intel P4 800 MHz

- **Chipset**
  - North Bridge
  - South Bridge

- **I/O Devices**
  - NICs
  - Disks
  - Displays
  - Keyboards

- **Current System Architecture:**
  - Isolated I/O: Separate memory (system) and I/O buses.

- **Important issue:** Which component creates a system performance bottleneck?

---

**Important note:**
- **System Architecture** consists of System Components and System Component Interconnects.
- **CPU Core** is equipped with Deep Instruction Pipelines and Dynamic scheduling.
- **SDRAM** is available in PC100/PC133 and DDR variants.
- **Caches** are categorized into L1, L2, and L3 with varying capacities and ways.
- **Main I/O Bus** supports various data rates and bit-widths.
- **I/O Subsystem** includes examples like Alpha, AMD K7, and Intel PII, PIII.
- The **current system architecture** is isolated I/O with separate memory and I/O buses.

---

**System Architecture = System Components + System Component Interconnects**

- **System Components** include CPU, Caches, Memory, etc.
- **System Component Interconnects** include buses and interconnects.

---

**Example:**
- **CPU Core:** 1 GHz - 3.8 GHz
- **Cache Levels:**
  - L1: 16-128K
  - L2: 256K-2M
  - L3: 2-16M
- **Memory Buses:**
  - PC100/PC133
  - DDR PC3200
  - RAMbus DRAM

---

**Important issues:**
- **Performance Bottleneck:** Identify which component limits system performance.
- **Interconnects:** Evaluate the impact of different buses and interfaces on system performance.

---

**References:**
- Shaaban Shaaban
- Fall 2009
- Lec #9
- Fall 2009
- 10-27-2009
Main Types of Buses/Interconnects in The System

1. **Processor-Memory Bus/Interconnect:**
   - Should offer very high-speed (bandwidth) and low latency.
   - Matched to the memory system performance to maximize memory-processor bandwidth.
   - Usually system design-specific (not an industry standard).
   - Examples: Alpha EV6 (AMD K7), Peak bandwidth = 400 MHz x 8 = 3.2 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
     HyperTransport 2.0: 200Mhz-1.4GHz, Peak bandwidth up to 22.8 GB/s

2. **I/O buses/Interconnects:**
   - Follow bus/interface industry 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 to processor-memory bus via a bus adapter (system chipset south bridge).
   - Examples: Main system I/O bus: PCI, PCI-X, PCI Express
     Storage Interfaces: SATA, PATA, SCSI.

System Architecture = System Components + System Component Interconnects
Intel Pentium 4 System Architecture
(Using The Intel 925 Chipset)

CPU
(Including cache)

System Bus (Front Side Bus, FSB)
Bandwidth usually should match or exceed that of main memory

System Memory
Two 8-byte DDR2 Channels

Memory Controller Hub
(Chipset North Bridge)

Graphics I/O Bus (PCI Express)

System Core Logic

I/O Controller Hub
(Chipset South Bridge)

Basic Input Output System (BIOS)

System Core Logic

I/O Subsystem

Main I/O Bus (PCI)
Misc. I/O Interfaces

Storage I/O (Serial ATA)

Misc.
I/O Interfaces

Current System Architecture:
Isolated I/O:  Separate memory and I/O buses.

## Bus Characteristics

<table>
<thead>
<tr>
<th>Option</th>
<th>High performance</th>
<th>Low cost/performance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bus width</strong></td>
<td>Separate address &amp; data lines</td>
<td>Multiplex address &amp; data lines</td>
</tr>
<tr>
<td><strong>Data width</strong></td>
<td>Wider is faster (e.g., 64 bits)</td>
<td>Narrower is cheaper (e.g., 16 bits)</td>
</tr>
<tr>
<td><strong>Transfer size</strong></td>
<td>Multiple words has less bus overhead</td>
<td>Single-word transfer is simpler</td>
</tr>
<tr>
<td><strong>Bus masters</strong></td>
<td>Multiple (requires arbitration)</td>
<td>Single master (no arbitration)</td>
</tr>
<tr>
<td><strong>Split</strong></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><strong>Clocking</strong></td>
<td>Synchronous</td>
<td>Asynchronous</td>
</tr>
</tbody>
</table>

FSB = Front Side Bus (Processor-memory Bus or System Bus)

(e.g. FSB)

#8  Lec # 9  Fall 2009  10-27-2009
### Example CPU-Memory System Buses

(Front Side 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>

FSB Bandwidth matched with single 8-byte channel SDRAM

FSB Bandwidth matched with dual channel PC3200 DDR SDRAM
## Main System I/O Bus Example: PCI, PCI-Express

<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>Legacy PCI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI 2.3</td>
<td>32</td>
<td>33.3</td>
<td>133</td>
</tr>
<tr>
<td>PCI 2.3</td>
<td>64</td>
<td>33.3</td>
<td>266</td>
</tr>
<tr>
<td>PCI 2.3</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>
<tr>
<td>Not Implemented Yet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI-X 2.0</td>
<td>64</td>
<td>266, 533</td>
<td>2100, 4200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formerly Intel’s 3GIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI-Express</td>
</tr>
</tbody>
</table>

### Addressing
- **Physical**
- **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>EIDE/Parallel ATA (PATA)</th>
<th>SCSI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Width</strong></td>
<td>16 bits</td>
</tr>
<tr>
<td><strong>Clock Rate</strong></td>
<td>Upto 100MHz</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bus Masters</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>Max no. devices</strong></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Peak Bandwidth</strong></td>
<td>200 MB/s</td>
</tr>
<tr>
<td><strong>Target Application</strong></td>
<td>Desktop</td>
</tr>
</tbody>
</table>

**EIDE** = Enhanced Integrated Drive Electronics  
**ATA** = Advanced Technology Attachment  
**PATA** = Parallel ATA  
**SATA** = Serial ATA  
**SCSI** = Small Computer System Interface

---

**EECC551 - Shaaban**

#11  Lec # 9  Fall 2009  10-27-2009
I/O Data Transfer Methods

1. **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.)

2. **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. (e.g. data is ready)
   - 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.
   - Used for moderate speed I/O (optical drives, storage, networks ..)
   - Allows overlap of CPU processing time and I/O processing time

\[
\text{Time(workload)} = \text{Time(CPU)} + \text{Time(I/O)} - \text{Time(Overlap)}
\]
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)
- Allows more overlap of CPU processing time and I/O processing time than interrupt-driven I/O.

**DMA transfer steps:**

1. 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.
2. The DMA controller starts the operation. When the data is available it transfers the data, including generating memory addresses for data to be transferred.
3. Once the DMA transfer is complete, the controller interrupts the processor, which determines whether the entire operation is complete.
I/O Interface/Controller

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

- Specific to each type of I/O device/interface standard.
- 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, USB ...)
- 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.
**I/O Controller Architecture**

- **Peripherals or Main I/O Bus (PCI, PCI-X, etc.)**
  - Host
  - Memory
  - Processor
  - Cache
  - Host Processor

- **Peripheral Bus Interface/DMA**
  - Buffer Memory
  - ROM
  - I/O Channel Interface
  - I/O Devices
  - SCSI, IDE, USB, ...
  - Industry-standard interfaces

**Equation:**

\[
\text{Time(workload)} = \text{Time(CPU)} + \text{Time(I/O)} - \text{Time(Overlap)}
\]

- **No overlap**
- **Overlap of CPU processing Time and I/O processing time**
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
    
    \[ \Rightarrow \text{ 5 times system performance increase} \]
    
    (50% loss in performance)
  
  - Increasing CPU performance by 100 times
    
    \[ \Rightarrow \text{ ~10 times system performance} \]
    
    (90% loss of performance)

- **The I/O system performance bottleneck diminishes the benefit of faster CPUs on overall system performance.**

System performance depends on many aspects of the system

("limited by weakest link in the chain"): The system performance bottleneck
System & I/O Performance Metrics/Modeling

- **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.

**I/O Performance Modeling:**

- **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 (or Entire System) Performance Metrics:**

1. **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 queue is never empty (server always busy).

2. **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 I/O device service time and buffer waiting (or queuing time).
   
   ⇒ Minimized when task queue is always empty (no queuing time).
   
   Response Time = Service Time + Queuing Time
System & I/O Performance Metrics: Throughput

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

- I/O Throughput is measured in two ways:
  
  1. **I/O rate:**
     - Measured in: I/O Tasks/sec
       - 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.

  2. **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 and multimedia applications.
System & I/O Performance Metrics: Response time

- Response time measures how long a storage (or I/O) system takes to process an I/O request and access data.
  - I/O request latency or total processing time per I/O request.

- 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.

The utilization of DMA and I/O device queues and multiple I/O devices servicing a queue may make throughput $\gg \frac{1}{\text{response time}}$.
**I/O Modeling:**

**Producer-Server Model**

- **Throughput:**
  - The number of tasks completed by the server in unit time.
  - In order to get the highest possible throughput:
    - The server should never be idle.
    - The queue should never be empty.

- **Response time:**
  - Begins when a task is placed in the queue.
  - Ends when it is completed by the server.
  - In order to minimize the response time:
    - The queue should be empty (no waiting time in queue).
    - The server will be idle at times.

Time system $= \frac{\text{Response Time}}{\text{Queue Time} + \text{Service Time}}$

**I/O Tasks**

- Producer (i.e., User, OS, or CPU)
- Queue (FIFO)
- Server (i.e., I/O device + controller)

**Throughput is maximized when:**
- The server should never be idle.
- The queue should never be empty.

**Response Time is minimized when:**
- The queue should be empty (no waiting time in queue).
- The server will be idle at times.
Response Time = Time_{System} = Time_{Queue} + Time_{Server} = T_q + T_{ser}

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
I/O Performance: Throughput Enhancement

In general throughput can be improved by:
- Throwing more hardware at the problem.
- Reduces load-related latency.

Response time is much harder to reduce.
- e.g. Faster I/O device (i.e server)

Response Time = Time_{System} = Time_{Queue} + Time_{Server} = T_q + T_{ser}
Magnetic Disks

**Characteristics:**

- **Diameter (form factor):** 1.8in - 3.5in
- **Rotational speed:** 5,400 RPM - 15,000 RPM
- **Tracks per surface.**
- **Sectors per track:** Outer tracks contain more sectors.
- **Recording or Areal Density:** Tracks/in \( \times \) Bits/in
- **Cost Per Megabyte.**
- **Seek Time:** (2-12 ms) Current Areal Density ~ 400 Gbits / Inch\(^2\)
  - 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. (~ time for half a rotation)
- **Transfer time:** The time needed to transfer a sector of bits.
- **Type of controller/interface:** SCSI, EIDE (PATA, SATA)
- **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)*

**Access time =** average seek time + average rotational delay
Basic 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 service time for a 500-byte (.5 KB) Sector?

\[
\text{Average service time} = \text{Average seek time} + \text{Average rotational delay} + \text{Transfer time} + \text{Controller overhead}
\]

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

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

Here: 

- 1KBytes = 10^3 bytes, 
- MByte = 10^6 bytes, 
- 1 GByte = 10^9 bytes
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.

Current Areal Density ~ 400 Gbits / In²
Internal data transfer rate increase is influenced by the increase in areal density over last 20 years.
Access/Seek Time is a big factor in service(response) time for small/random disk requests. Limited improvement due to mechanical rotation speed + seek delay.

Less than 3x times improvement over 15 years!
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.

Historic Perspective of Hard Drive Characteristics Evolution: Cost

Cost Per MByte: > 100,000X times cost drop

Actual Current Hard Disk Storage Cost (Third Quarter 2009): 0.0001 dollars per MByte or about 10 GBytes /Dollar
Historic Perspective of Hard Drive Characteristics Evolution: **Roadmap**

Current Areal Density $\sim 400$ Gbits / In$^2$
Introduction to Queuing Theory

(Steady State)

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

  Rate $r$

  

- **Little’s Law:**
  
  $$L_{sys} \text{ (length or number of tasks in system)} = r \text{ (arrival rate)} \times T_{sys} \text{ (System Time)}$$

  Mean number tasks in system

- 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

System (Single Queue + Single Server)

- 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 $T_q$ thus $T_{sys} = T_{ser} + T_q$
  - $r$: Average number of arriving tasks/sec (i.e. task arrival rate)
  - $L_{ser}$: 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:
- $L_{sys} = r \times T_{sys}$ (applied to system)
- $L_q = r \times T_q$ (applied to queue)

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

$u$ must be between 0 and 1 otherwise there would be more tasks arriving than could be serviced.
A Little Queuing Theory

- **Server spends a variable amount of time with customers**
  - Arithmetic mean time = $m_1 = (f_1 \times T_1 + f_2 \times T_2 + \ldots + f_n \times T_n)$
    - where $T_i$ is the time for task $i$ and $f_i$ is the frequency of task $i$
  - variance = $(f_1 \times T_1^2 + f_2 \times T_2^2 + \ldots + f_n \times T_n^2) - m_1^2$
  - Must keep track of unit of measure (100 ms$^2$ vs. 0.1 s$^2$)
    - *Squared coefficient of variance*: $C^2 = \text{variance}/m_1^2$
    - Unitless measure

- **Distributions:**
  - **Exponential (Poisson) distribution** $C^2 = 1$: most short relative to average, few others long; 90% $< 2.3 \times$ average, 63% $< \text{average}$
  - **Hypoexponential distribution** $C^2 < 1$: most close to average, $C^2 = 0.5 \Rightarrow 90\% < 2.0 \times \text{average}$, only 57% $< \text{average}$
  - **Hyperexponential distribution** $C^2 > 1$: further from average, $C^2 = 2.0 \Rightarrow 90\% < 2.8 \times \text{average}$, 69% $< \text{average}$

$$\text{Variance} = (\text{Standard deviation})^2$$
A Little Queuing Theory

- Service time completions vs. waiting time for a busy server: randomly arriving task 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 task at a time (server) + waiting area (queue): together called a system
- Server spends a variable amount of time servicing tasks, average, \( T_{\text{system}} = T_{\text{queue}} + T_{\text{server}} \)

\[
T_{\text{queue}} = \text{Length}_{\text{queue}} \times T_{\text{server}} + \text{Time for the server to complete current task}
\]

\[
\text{Time for the server to complete current task} = \text{Server utilization} \times \text{remaining service time of current task}
\]

\[
\text{Length}_{\text{queue}} = \text{Arrival Rate} \times \text{Time}_{\text{queue}} \quad (\text{Little’s Law})
\]

We need to estimate waiting time in queue (i.e. \( T_{\text{queue}} = T_q \))? \( T_q \)?

Here a server is the device (i.e hard drive) and its I/O controller (IOC)
The response time above does not account for other factors such as CPU time.
A Little Queuing Theory: Average Queue Wait Time $T_q$

- Calculating average wait time in queue $T_q$
  - If something at server, it takes to complete on average $m1(z) = 1/2 \times T_{ser} \times (1 + C^2)$
  - Chance server is busy = $u$; average delay is $u \times m1(z) = 1/2 \times u \times T_{ser} \times (1 + C^2)$
  - All customers in line must complete; each avg $T_{ser}$

$$T_{queue} = \text{Time for the server to complete current task} + \text{Length}_{queue} \times \text{Time}_{server}$$

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

Rearrange:

$$L_q = r \times T_q$$

(Little’s Law)

- Notation:
  - $r$: average number of arriving tasks/second
  - $T_{ser}$: average time to service a task
  - $u$: server utilization (0..1): $u = r \times T_{ser}$
  - $T_q$: average time/request in queue
  - $L_q$: average length of queue: $L_q = r \times T_q$

What if utilization $u = 1$ ?

A version of this derivation in textbook page 385 (3rd Edition: page 726)
A Little Queuing Theory: M/G/1 and M/M/1

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

- Described “memoryless” or Markovian request arrival (M for $C^2 = 1$ exponentially random), General service distribution (no restrictions), 1 server: **M/G/1 queue**

- When Service times have $C^2 = 1$, **M/M/1 queue**
  
  \[ T_q = T_{ser} \times u \times \frac{1 + C^2}{2(1 - u)} = \frac{T_{ser} \times u}{1 - u} \]

- Queuing Time, $T_q$

- Average time/task in queue

- Average time to service a task

- Average length of queue

- Server utilization (0..1)

- System response time:

\[ T_{sys} = T_{queue} + T_{server} = T_{sys} = T_q + T_{ser} \]
Single Queue + Multiple Servers (Disks/Controllers)

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 = \( 1/ T_{ser} \)
  (and requests are distributed evenly among all servers)

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

where \( u = r \times T_{ser} / m \)

- \( m \) number of servers
- \( T_{ser} \) average time to service a task
- \( u \) server utilization (0..1): \( u = r \times T_{ser} / m \)
- \( T_q \) average time/task in queue
- \( L_q \) Average length of queue \( L_q = r \times T_q \)
- \( T_{sys} = T_{ser} + T_q \) Time in system (mean response time)

Please Note:
We will use this simplified formula for M/M/m not the book version 4th Edition on page 388
(3rd Edition: page729)

i.e as if the \( m \) servers are a single server with an effective service time of \( T_{ser} / m \)
I/O Queuing Performance: An M/M/1 Example

- A processor sends 40 disk I/O requests per second, requests & service are exponentially distributed, average disk service time = 20 ms

- On average:
  - What is the disk utilization \( u \)?
  - 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 = 40
  - \( T_{ser} \) average time to service a request = 20 ms (0.02s)

- We obtain:
  - \( u \) server utilization: \( u = r \times T_{ser} = 40/s \times 0.02s = 0.8 \) or 80%
  - \( T_q \) average time/request in queue: \( T_q = T_{ser} \times u / (1 - u) \)
    = \( 20 \times 0.8/(1-0.8) = 20 \times 0.8/0.2 = 20 \times 4 = 80 \) ms (0.08s)
  - \( T_{sys} \) average time/request in system: \( T_{sys} = T_q + T_{ser} = 80 + 20 = 100 \) ms
  - \( L_q \) average length of queue: \( L_q = r \times T_q \)
    = \( 40/s \times 0.08s = 3.2 \) requests in queue
  - \( L_{sys} \) average # tasks in system: \( L_{sys} = r \times T_{sys} = 40/s \times 0.1s = 4 \)
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 40 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 = 40
- \( T_{ser} \) average time to service a request = 10 ms (0.01s)

We obtain:
- Server utilization: \( u = r \times T_{ser} = 40/s \times 0.01s = 0.4 \) or 40%
- Average time/request in queue: \( T_q = T_{ser} \times u / (1 - u) \)
  \( = 10 \times 0.4 / (1 - 0.4) = 10 \times 0.4 / 0.6 = 6.67 \) ms (0.0067s)
- Average time/request in system: \( T_{sys} = T_q + T_{ser} = 10 + 6.67 = 16.67 \) ms

Response time is \( 100/16.67 = 6 \) times faster even though the new service time is only 2 times faster due to lower queuing time.

6.67 ms instead of 80 ms
Factors Affecting System & I/O Performance

- **I/O processing computational requirements:**
  - CPU computations available for I/O operations.
  - Operating system I/O processing policies/routines.
  - I/O Data Transfer/Processing 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. i.e number of devices, array level ..
  - Loading level (u) of I/O devices (queuing delay, response time).

- **Memory subsystem performance:**
  - Available memory bandwidth for I/O operations (For DMA)

- **Operating System Policies:**
  - File system vs. Raw I/O.
  - File cache size and write Policy.
  - File pre-fetching, etc.

System performance depends on many aspects of the system
(“limited by weakest link in the chain”): The system performance bottleneck

Components of Total System Execution Time:

- CPU
- Memory
- I/O
System Design (Including I/O)

- When designing a system, the performance of the components that make it up should be balanced.

- Steps for designing I/O systems are:
  - List types and performance of I/O devices and buses in the system
  - Determine target application computational & I/O demands
  - Determine the CPU resource demands for I/O processing
    - 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)
  - Determine 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.
    - Improve performance of the component that poses a system performance bottleneck

System performance depends on many aspects of the system ("limited by weakest link in the chain")
Example: Determining the System Performance Bottleneck (ignoring I/O 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 (100 disks total)

• Other assumptions
  – All devices/system components can be used to 100% utilization
  – Average I/O request size is 16 KB
  – I/O Requests are assumed spread evenly on all disks.
  – OS uses 10,000 CPU instructions to process a disk I/O request
  – Ignore disk/controller queuing delays.
    (Since I/O queuing delays are ignored here 100% disk utilization is allowed)

• What is the average IOPS?
• What is the average I/O bandwidth?
• What is the average response time per IO operation?
Example: Determining the System I/O Bottleneck
(ignoring queuing delays)

• The performance of I/O systems is determined by the system component with the lowest performance (the system performance bottleneck):
  - **CPU**: \( \frac{500 \text{ MIPS}}{10,000 \text{ instructions per I/O}} = 50,000 \text{ IOPS} \)
    
    CPU time per I/O = \( \frac{10,000}{500,000,000} = 0.02 \text{ ms} \)
  - **Main Memory**: \( \frac{16 \text{ bytes}}{100 \text{ ns} \times 16 \text{ KB per I/O}} = 10,000 \text{ IOPS} \)
    
    Memory time per I/O = \( \frac{1}{10,000} = 0.1 \text{ ms} \)
  - **I/O bus**: \( \frac{200 \text{ MB/sec}}{16 \text{ KB per I/O}} = 12,500 \text{ IOPS} \)
  - **SCSI-2**: \( \frac{20 \text{ buses}}{1 \text{ ms} + \frac{16 \text{ KB}}{20 \text{ MB/sec}}} = 11,111 \text{ IOPS} \)
    
    SCSI bus time per I/O = \( 1 \text{ ms} + \frac{16}{20} = 1.8 \text{ ms} \)
  - **Disks**: \( \frac{100 \text{ disks}}{8 \text{ ms} + \frac{0.5}{7200 \text{ RPMS}} + \frac{16 \text{ KB}}{6 \text{ MB/sec}}} = 6700 \text{ IOPS} \)

  \[ T_{\text{disk}} = 8 + 4.2 + 2.7 = 14.9 \text{ ms} \]

• The disks limit the I/O performance to  **6700 IOPS**

• The average I/O bandwidth is  **6700 IOPS \times (16 \text{ KB/sec}) = 107.2 \text{ MB/sec}**

• **Response Time Per I/O = Tcpu + Tmemory + Tscsi + Tdisk =**
  
  \[ = 0.02 + 0.1 + 1.8 + 14.9 = 16.82 \text{ ms} \]

Since I/O queuing delays are ignored here 100% disk utilization is allowed.
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 (100 disks)

• Other assumptions
  – All devices used to 60% utilization (i.e. u = 0.6).
  – Treat the I/O system as an M/M/m queue.
  – I/O Requests are assumed spread evenly on all disks.
  – Average I/O size is 16 KB
  – OS uses 10,000 CPU instructions to process a disk I/O request

• What is the average IOPS? What is the average bandwidth?
• Average response time per IO operation?

Here: \( m = 100 \)

Here: 1KBytes = 10\(^3\) bytes, MByte = 10\(^6\) bytes, 1 GByte = 10\(^9\) bytes
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 performance (the system performance bottleneck):
  - CPU: \((500 \text{ MIPS})/(10,000 \text{ instr. per I/O}) \times 0.6 = 30,000 \text{ IOPS}\)
    
    \[
    \text{CPU time per I/O} = \frac{10,000}{500,000,000} = 0.02 \text{ ms}
    \]
  - Main Memory: \((16 \text{ bytes})/(100 \text{ ns} \times 16 \text{ KB per I/O}) \times 0.6 = 6,000 \text{ IOPS}\)
    
    \[
    \text{Memory time per I/O} = \frac{1}{10,000} = 0.1 \text{ ms}
    \]
  - I/O bus: \((200 \text{ MB/sec})/(16 \text{ KB per I/O}) \times 0.6 = 12,500 \text{ IOPS}\)
    
    \[
    \text{I/O bus time per I/O} = \frac{1}{16/20} \text{ ms} = 1.8 \text{ ms}
    \]
  - SCSI-2: \((20 \text{ buses})/((1 \text{ ms} + (16 \text{ KB})/(20 \text{ MB/sec})) \times 0.6 = 6,666.6 \text{ IOPS}\)
    
    \[
    \text{SCSI bus time per I/O} = 1 \text{ ms} + 0.8 \text{ ms} = 1.8 \text{ ms}
    \]
  - Disks: \((100 \text{ disks})/((8 \text{ ms} + 0.5/(7200 \text{ RPM}) + (16 \text{ KB})/(6 \text{ MB/sec})) \times 0.6 = 6,700 \text{ x} 0.6 = 4020 \text{ IOPS}\)
    
    \[
    T_{ser} = (8 \text{ ms} + 0.5/(7200 \text{ RPM}) + (16 \text{ KB})/(6 \text{ MB/sec}) = 8 + 0.2 + 0.27 = 8.47 \text{ ms}
    \]

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

Example: Determining the I/O Bottleneck

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

Throughput

Using expression for \( T_q \) for M/M/m from slide 36

Total System response time including CPU time and other delays

Here: \(1 \text{ KBytes} = 10^3 \text{ bytes}, \ 1 \text{ MByte} = 10^6 \text{ bytes}, \ 1 \text{ GByte} = 10^9 \text{ bytes} \)