Input/Output & System Performance Issues

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

Isolated I/O System Architecture

- **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)**

More Specifically steady state queuing theory

Quiz 8

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:**

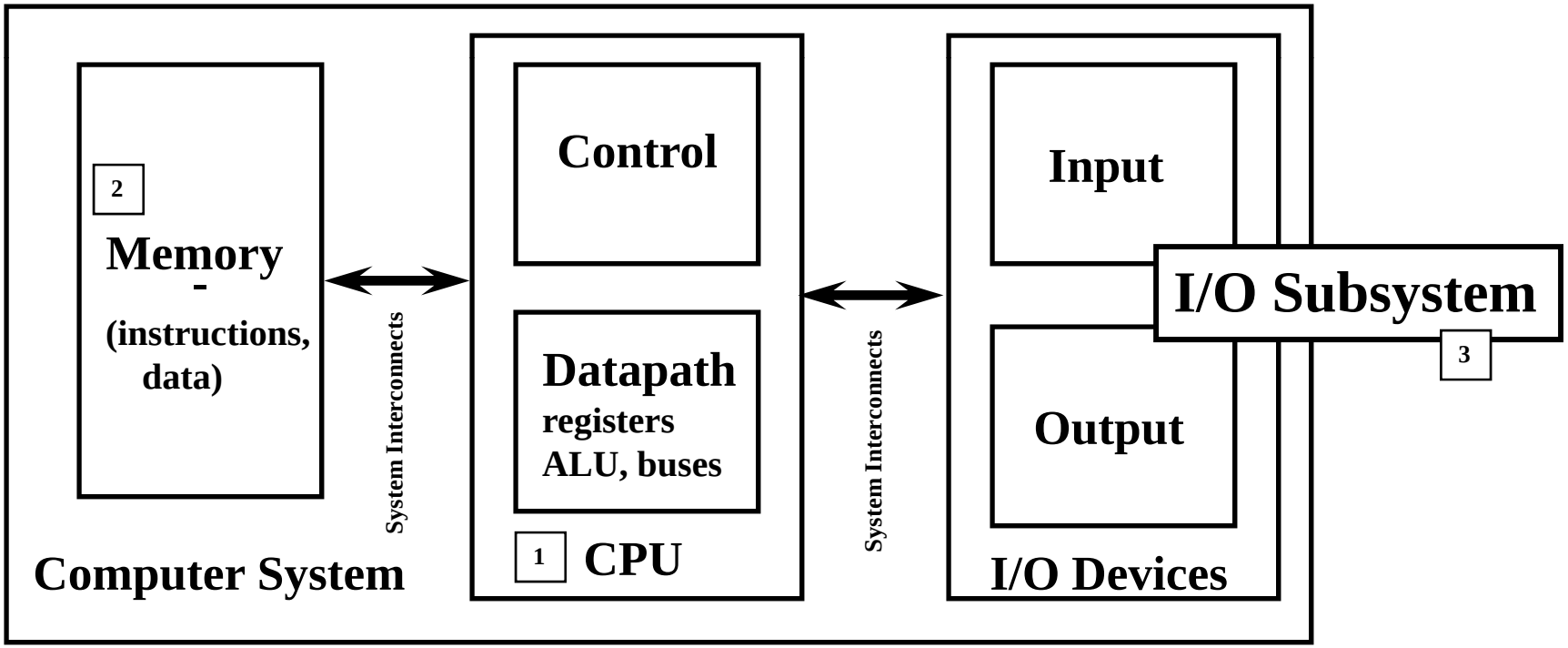
1 – **Central Processing Unit (CPU):** Control Unit (instruction decode, sequencing of operations), Datapath (registers, arithmetic and logic unit, buses).

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)

Components of Total System Execution Time: (or response time) CPU | Memory | I/O



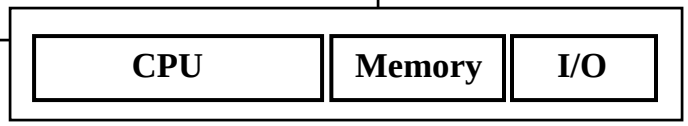
System performance depends on many aspects of the system ("limited by weakest link in the chain"): The system performance bottleneck

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).**
  - CPU: Including memory
  - I/O devices: Including users

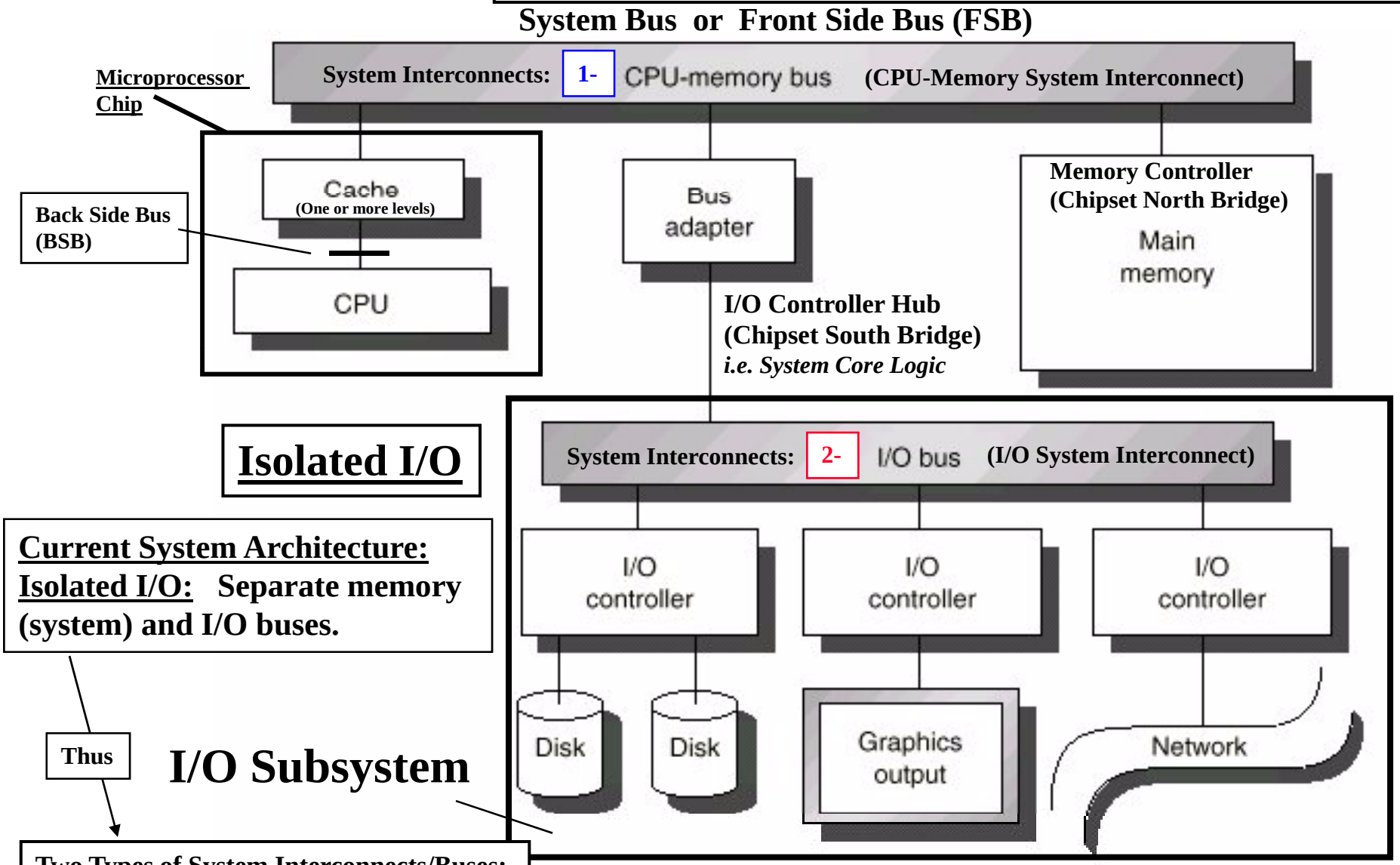
- **Design factors:**
  - **I/O device characteristics (input, output, storage, etc.) /Performance.**
  - **I/O Connection Structure (degree of separation from memory operations).** → **Isolated I/O System Architecture**
  - **I/O interface (the utilization of dedicated I/O and bus controllers).**
  - **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)



Typical FSB-Based System Architecture

System Architecture = System Components + System Component Interconnects



Isolated I/O

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

Thus

I/O Subsystem

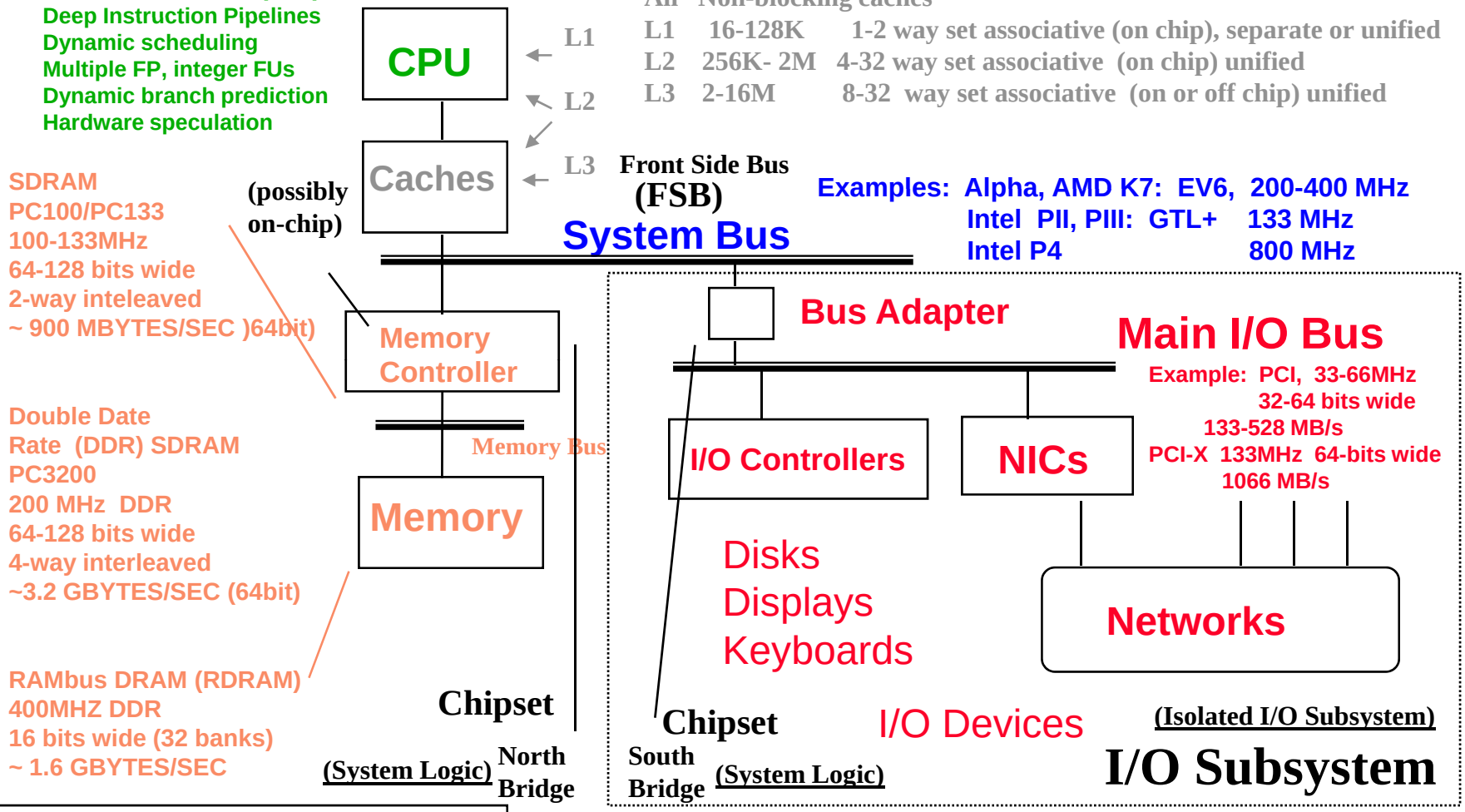
Two Types of System Interconnects/Buses:
1- CPU-Memory Bus or interconnect
2 – I/O Buses/interfaces

Typical FSB-Based 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

All Non-blocking 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



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

Thus →

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

Important issue: Which component creates a system performance bottleneck?

Main Types of Buses/Interconnects in The System

1 Processor-Memory Bus/Interconnect:

AKA System Bus, Front Side Bus, (FSB)

- 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
  (point-to-point system interconnect not a bus)

Also Intel's QuickPath Interconnect (QPI) used in Core i7 system architecture

2 I/O buses/Interconnects:

Sometimes called I/O *channels or interfaces*

- 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).

Isolated I/O System Architecture

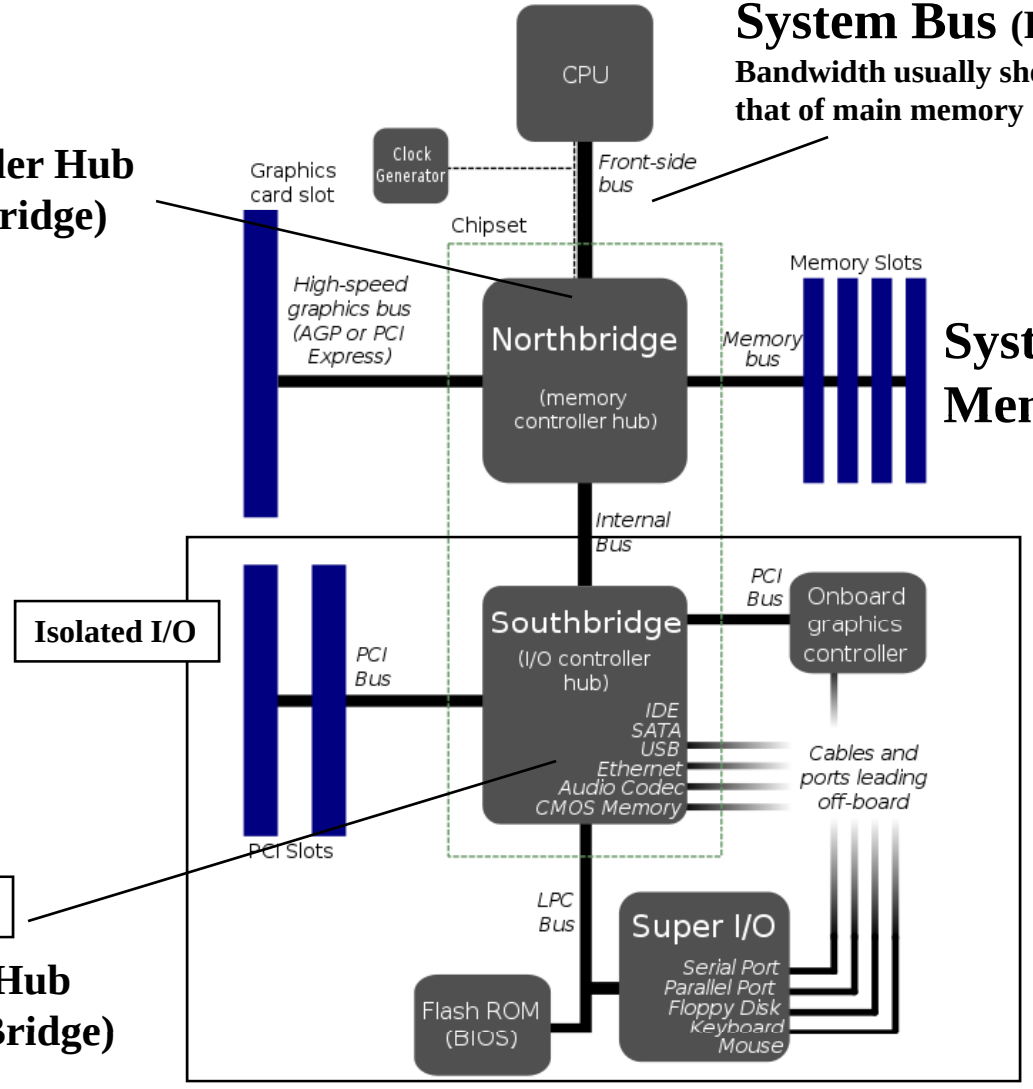
- Examples: Main system I/O bus: PCI, PCI-X, PCI Express
  Storage Interfaces: SATA, PATA, SCSI.

System Architecture = System Components + System Component Interconnects

FSB-Based Single Processor Socket System Architecture

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

System Core Logic
Memory Controller Hub (Chipset North Bridge)



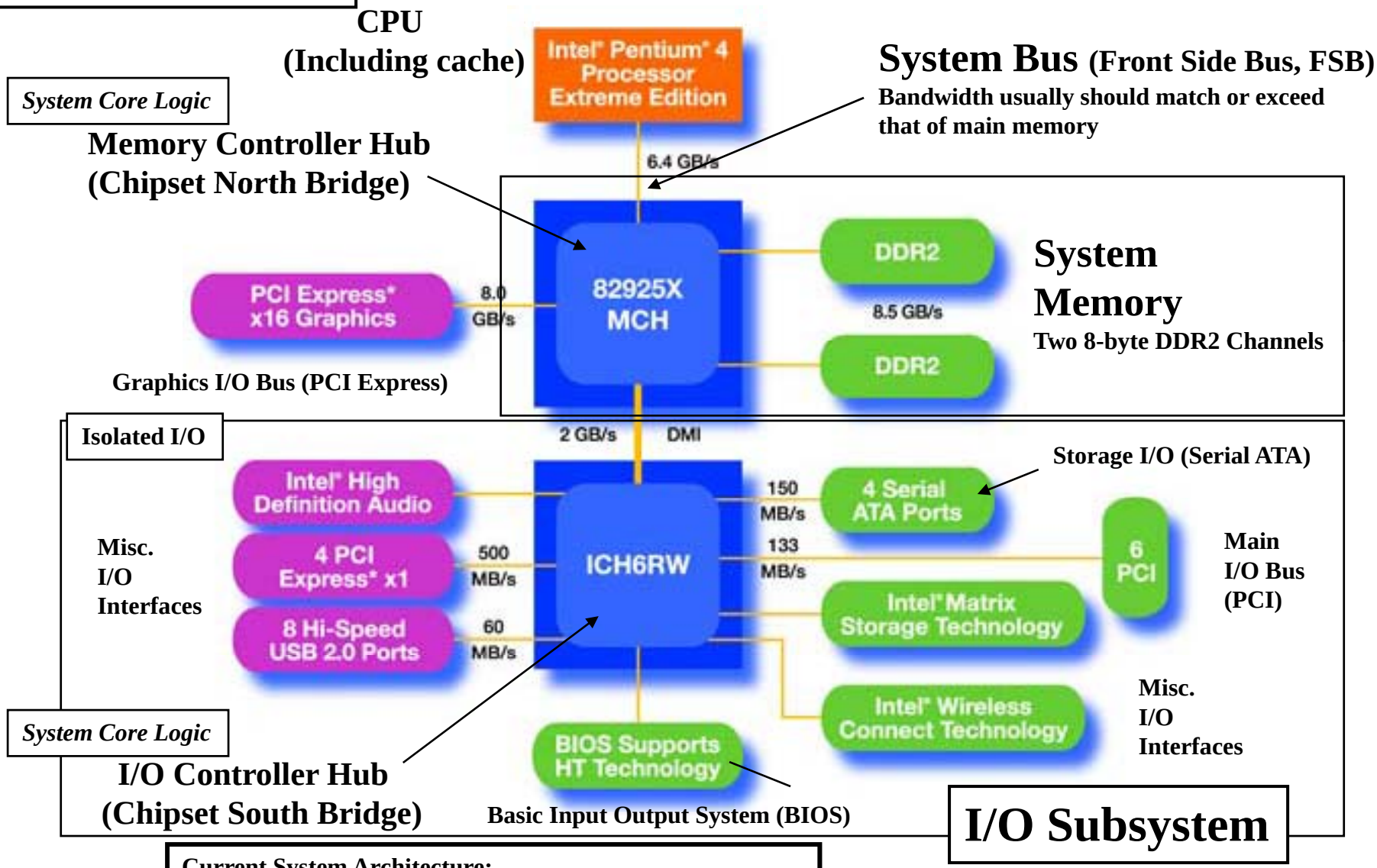
System Memory

Isolated I/O

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

Intel Pentium 4 System Architecture
(Using The Intel 925 Chipset)

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

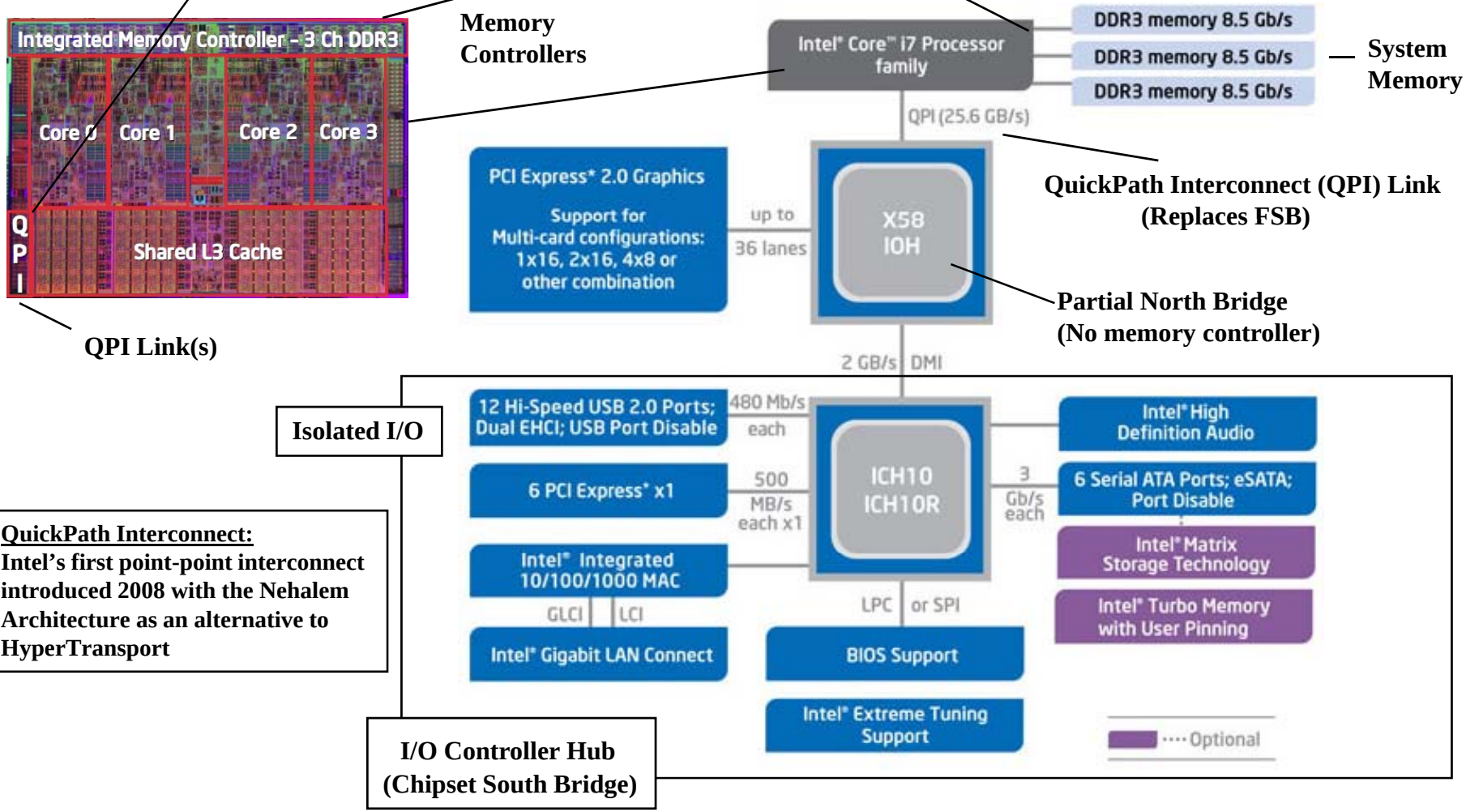


Current System Architecture:
Isolated I/O: Separate memory and I/O buses.

Source: http://www.anandtech.com/showdoc.aspx?i=2088&p=4

Intel Core i7 “Nehalem” System Architecture

Intel's QuickPath Interconnect (QPI) Point-to-point system interconnect used instead of Front Side Bus (FSB)
+ Memory controller integrated on processor chip (three DDR3 channels)



QuickPath Interconnect:
Intel’s first point-point interconnect introduced 2008 with the Nehalem Architecture as an alternative to HyperTransport

(e.g . FSB)

Bus Characteristics

Option	*High performance*	*Low cost/performance*
Bus width	Separate address & data lines	Multiplex address & data lines
Data width	Wider is faster (e.g., 64 bits)	Narrower is cheaper (e.g., 16 bits)
Transfer size	Multiple words has less bus overhead	Single-word transfer is simpler
Bus masters	Multiple (requires arbitration)	Single master (no arbitration)
Split	Yes, separate Request and Reply packets gets higher bandwidth (needs multiple masters)	No , continuous transaction? connection is cheaper and has lower latency
Clocking	Synchronous	Asynchronous

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

Example CPU-Memory System Buses (Front Side Buses, FSBs)

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

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

	Specification	Bus Width (bits)	Bus Frequency (MHz)	Peak Bandwidth (MB/sec)
Legacy PCI	PCI 2.3	32	33.3	133
Legacy PCI	PCI 2.3	64	33.3	266
Legacy PCI	PCI 2.3	64	66.6	533
Legacy PCI	PCI-X 1.0	64	133.3	1066
Not Implemented Yet	PCI-X 2.0	64	266, 533	2100 , 4200
Formerly Intel's 3GIO	PCI-Express	1-32	???	500-16,000

Addressing **Physical**

Master **Multi**

Arbitration **Central**

PCI Bus Transaction Latency:

PCI requires 9 cycles @ 33Mhz (272ns)
PCI-X requires 10 cycles @ 133MHz (75ns)

PCI = Peripheral Component Interconnect

Storage IO Interfaces/Buses

	EIDE/Parallel ATA (PATA)	SCSI
Data Width	16 bits	8 or 16 bits (wide)
Clock Rate	Upto 100MHz	10MHz (Fast) 20MHz (Ultra) 40MHz (Ultra2) 80MHz (Ultra3) 160MHz (Ultra4)
Bus Masters	1	Multiple
Max no. devices	2	7 (8-bit bus) 15 (16-bit bus)
Peak Bandwidth	200 MB/s	320MB/s (Ultra4)
Target Application	Desktop	Servers____

EIDE = Enhanced Integrated Drive Electronics
ATA = Advanced Technology Attachment
PATA = Parallel ATA
SATA = Serial ATA

SCSI = Small Computer System Interface

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.)

Memory-mapped register

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, neworks ..)
- Allows overlap of CPU processing time and I/O processing time

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

No overlap

Overlap of CPU processing
Time and I/O processing time

I/O data transfer methods:

3 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.**
- **Even lower 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 ...)**

Industry-standard interfaces

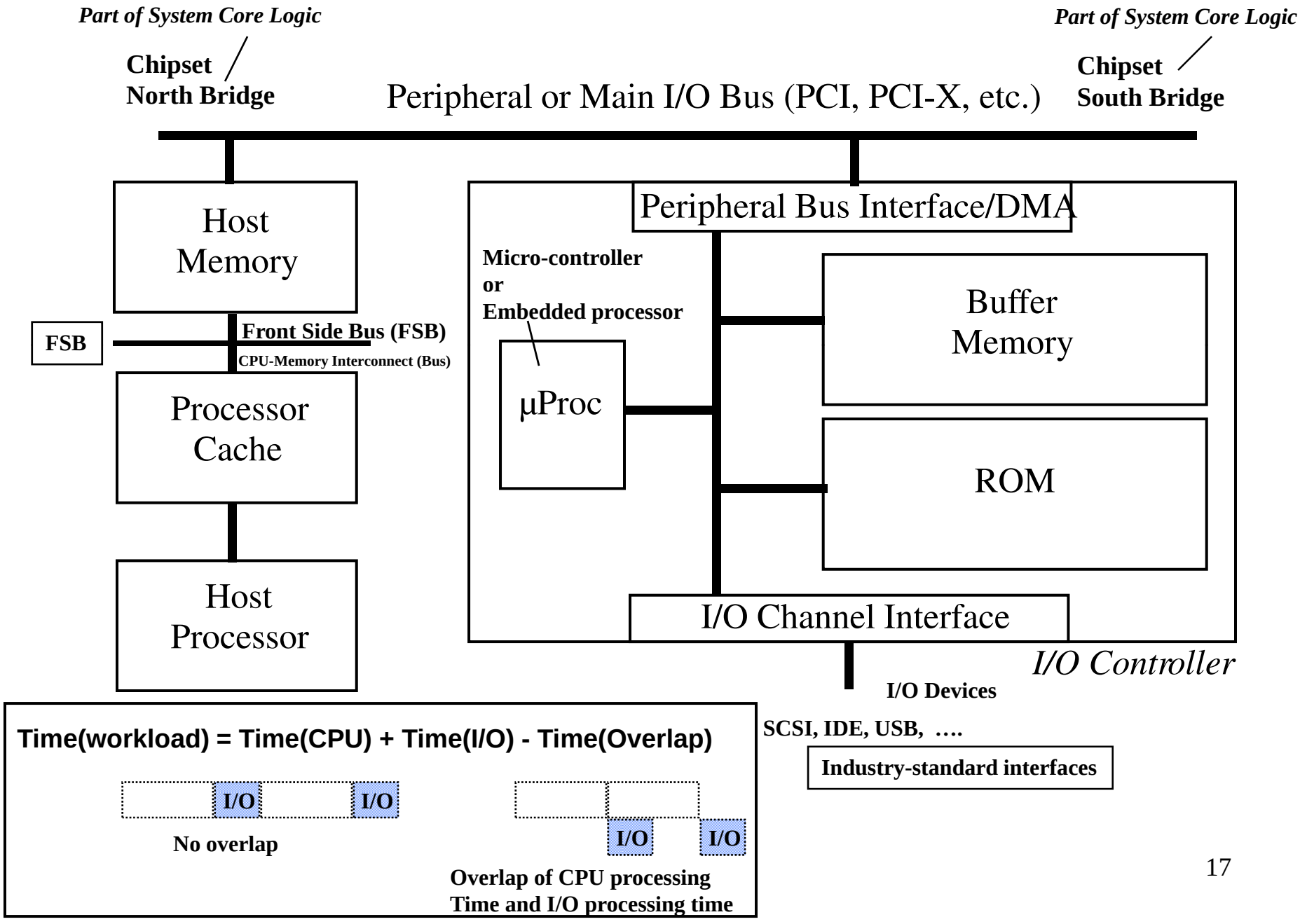
Why?

- **Handles I/O details (originally done by CPU) such as:**

Low-level I/O Processing off-loaded from CPU

- **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.**

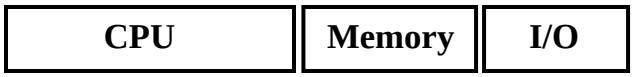
I/O Controller Architecture



I/O: A System Performance Perspective

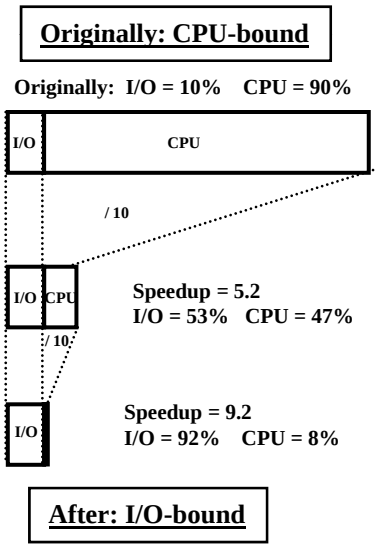
- **CPU Performance: Improvement of ~ 60% per year.**
- **I/O Sub-System Performance: Limited by *mechanical* delays (disk I/O). Improvement less than 10% per year (IO rate per sec or MB per sec).**
- **From Amdahl's Law: overall system speed-up is limited by the slowest component:**

i.e storage devices (hard drives)



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)**



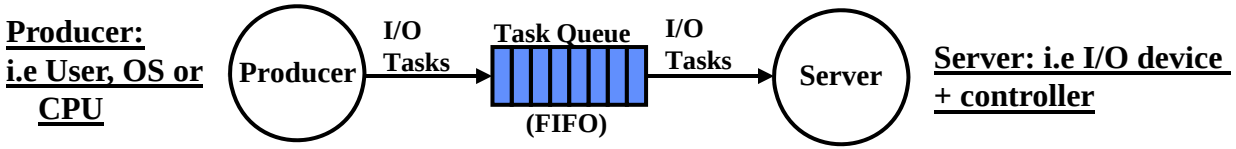
- **The I/O system performance bottleneck diminishes the benefit of faster CPUs on overall system performance.**

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 serice 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.** **i.e server applications**

2 • **Data rate, measured in *bytes/second* or *megabytes/second (MB/s, GB/s …*).**

  – **Data rate is generally used for applications where the size of each request is large, such as in scientific and multimedia applications.**

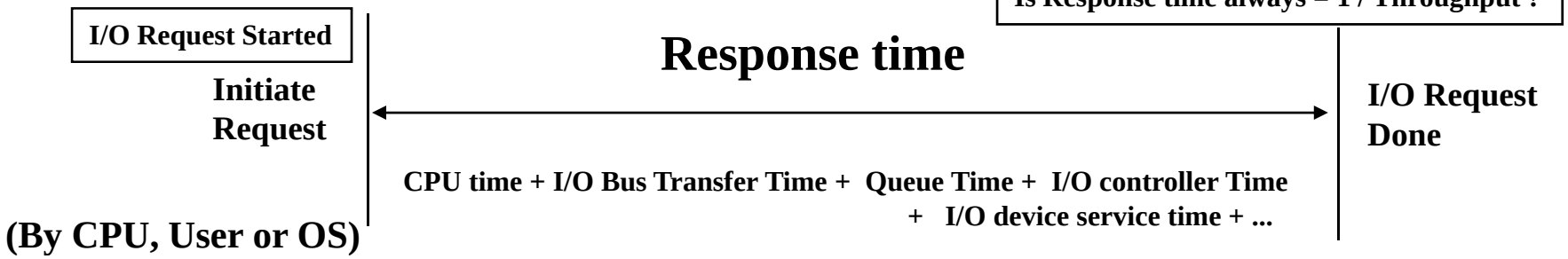
System & I/O Performance Metrics: Response time

- **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.**

Or entire system

i.e. Time it takes the system to process an average task

Is Response time always = 1 / Throughput ?

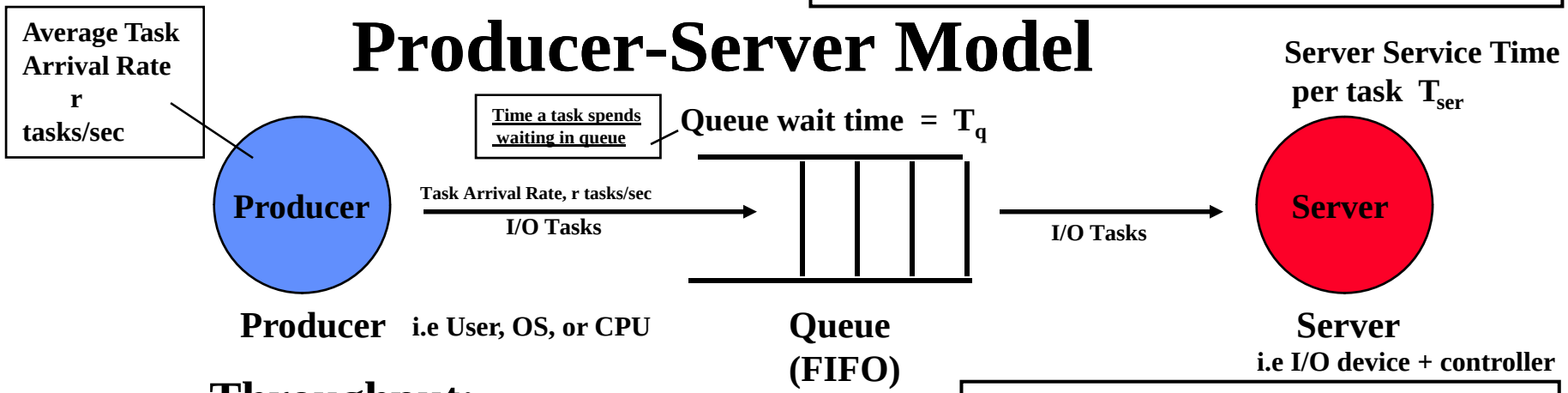


The utilization of DMA and I/O device queues and multiple I/O devices servicing a queue may make throughput >> 1 / response time

I/O Modeling:

$Time_{system}$ =Time in System for a task =
Response Time = Queuing Time + Service Time

Producer-Server Model



Shown above: Single Queue + Single Server

- **Throughput:**
  - The number of tasks completed by the server in unit time.
  - In order to get the highest possible throughput:

Throughput is maximized when:

- 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:

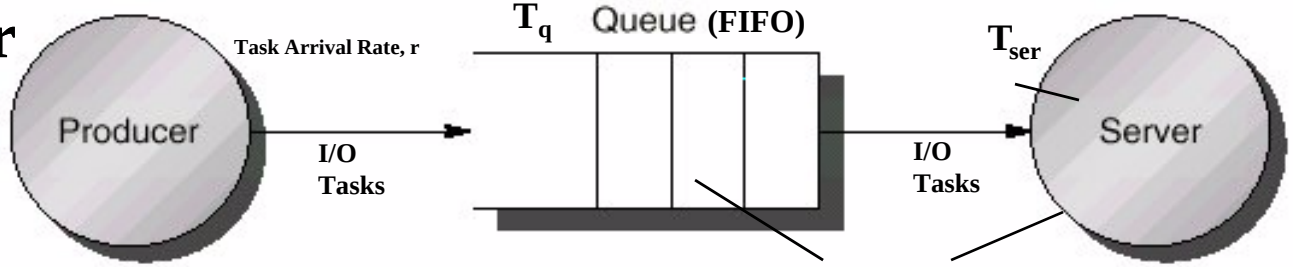
Response Time is minimized when:

- The queue should be empty (no waiting time in queue).
- The server will be idle at times.

Shown above is a (Single Queue + Single Server) Producer-Server Model

Producer-Server Model

Single Queue + Single Server

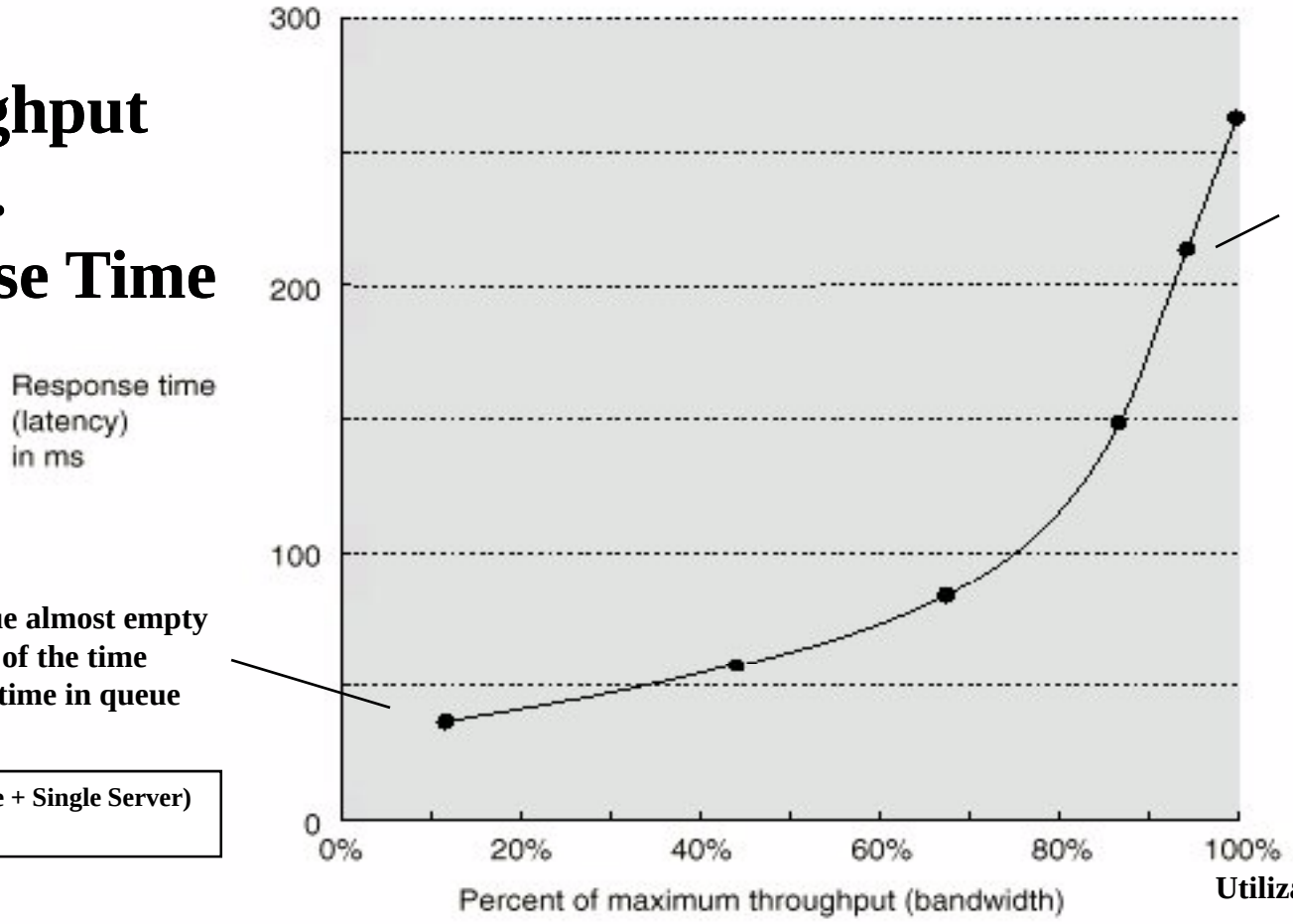


User or CPU

I/O device + controller

$$\text{Response Time} = \text{Time}_{System} = \text{Time}_{Queue} + \text{Time}_{Server} = T_q + T_{ser}$$

Throughput vs. Response Time



Queue full most of the time. More time in queue

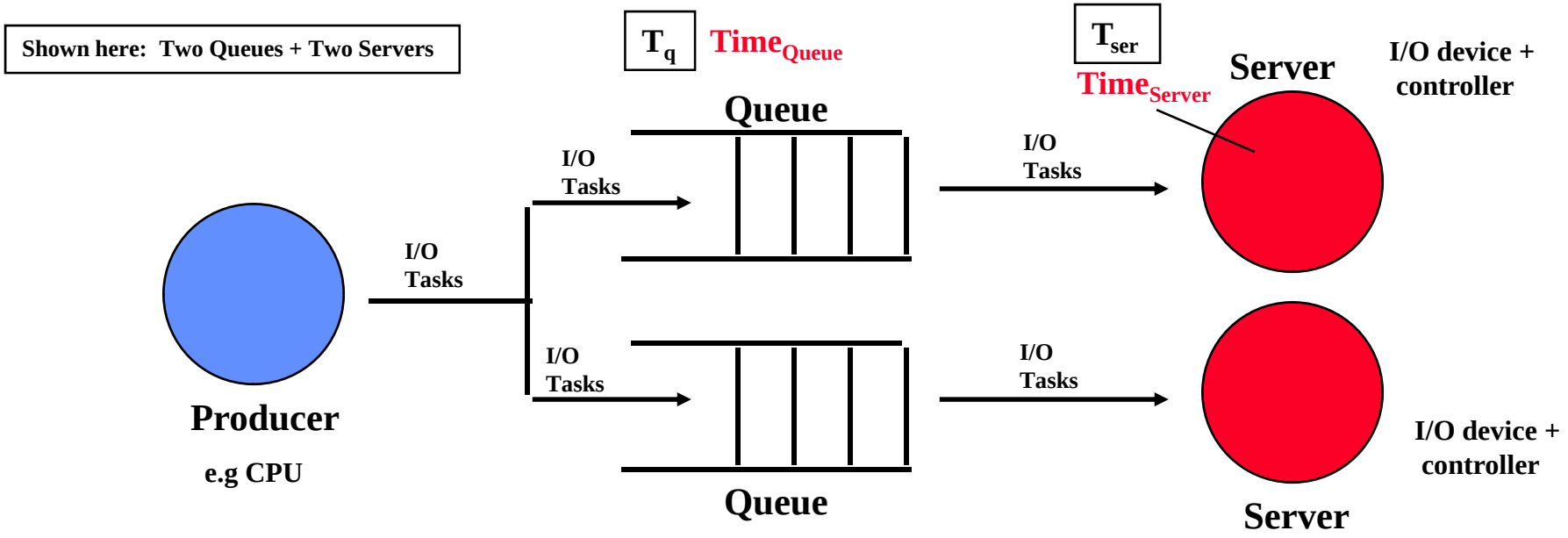
Queue almost empty most of the time
Less time in queue

Shown here is a (Single Queue + Single Server) Producer-Server Model

AKA Loading Factor

i.e Utilization = U ranges from 0 to 1 (0 % to 100%)

I/O Performance: Throughput Enhancement



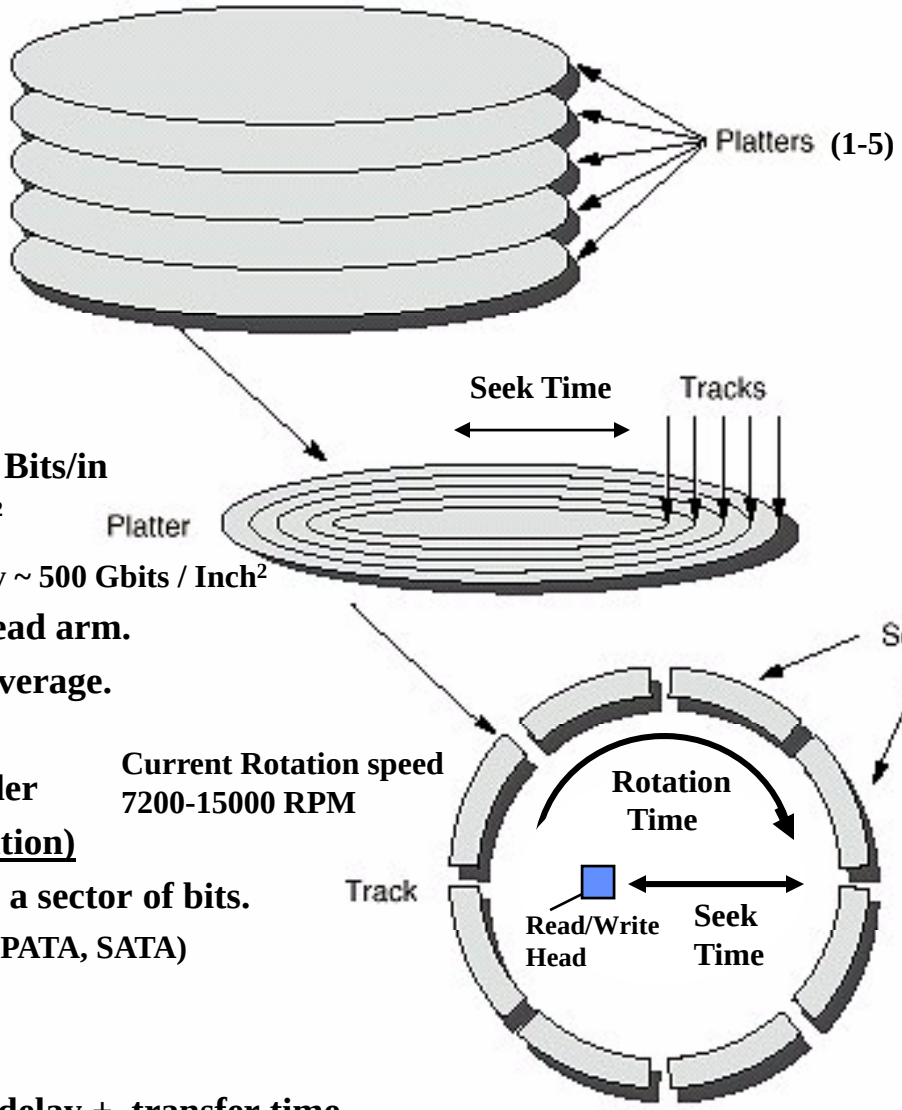
- In general throughput can be improved by:
  - Throwing more hardware at the problem.
  - Reduces load-related latency. Less queuing time
- Response time is much harder to reduce.
  - e.g. Faster I/O device (i.e server)

Ignoring CPU I/O processing time and other system delays

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 X Bits/in

  Bits/ Inch^2

  Current Areal Density ~ 500 Gbits / Inch^2
- 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. (~ 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?**

Time for half a rotation

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

= 5 ms + 0.5/(10000 RPM/60) + 0.5 KB/40 MB/s + 0.1 ms

= 5 + 3 + 0.13 + 0.1 = 8.23 ms

Access Time

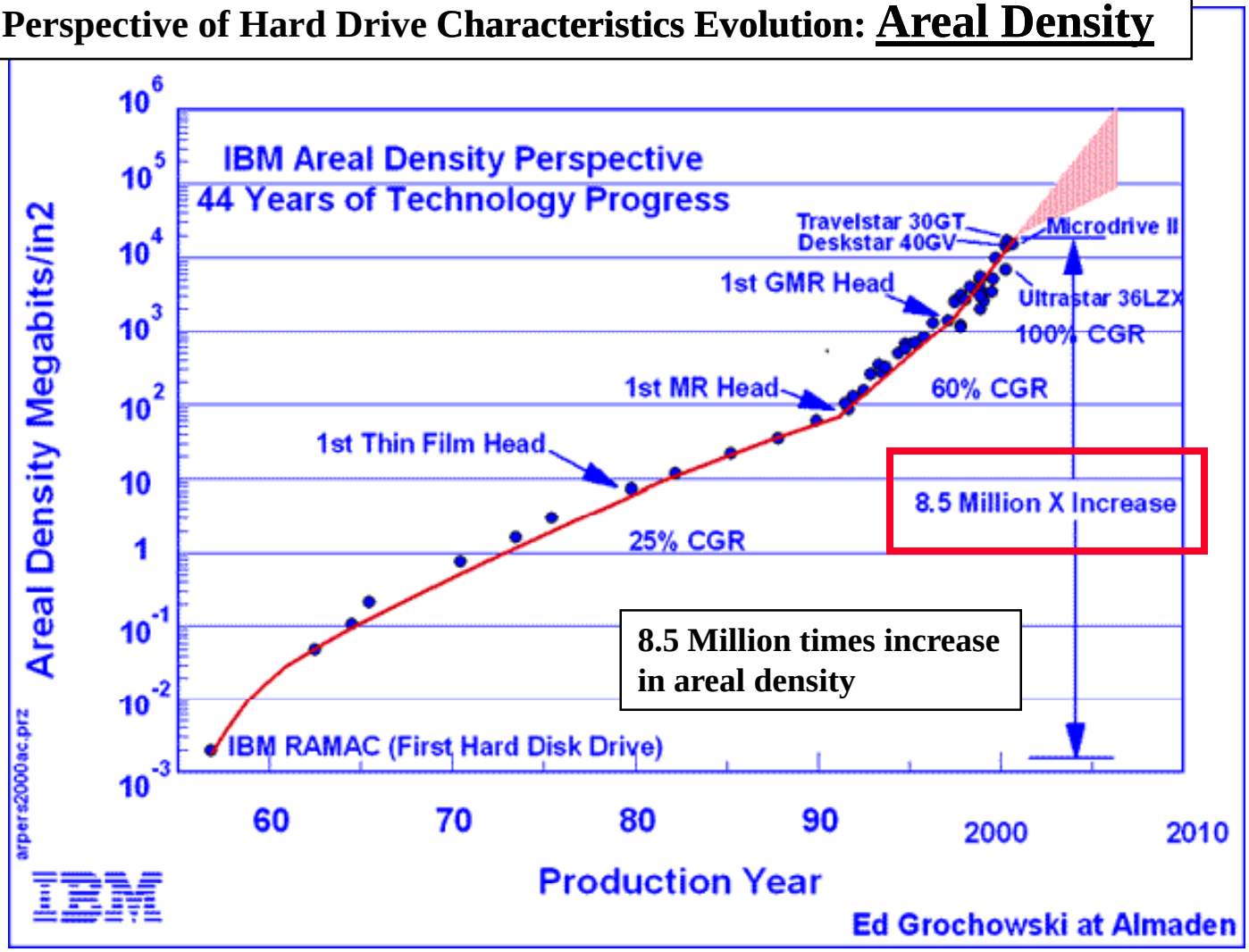
Actual time to process the disk request is greater and may include CPU I/O processing Time and queuing time

$T_{service}$ **(Disk Service Time for this request)**

Or T_{ser}

Here: 1KBytes = 10^3 bytes, MByte = 10^6 bytes, 1 GByte = 10^9 bytes

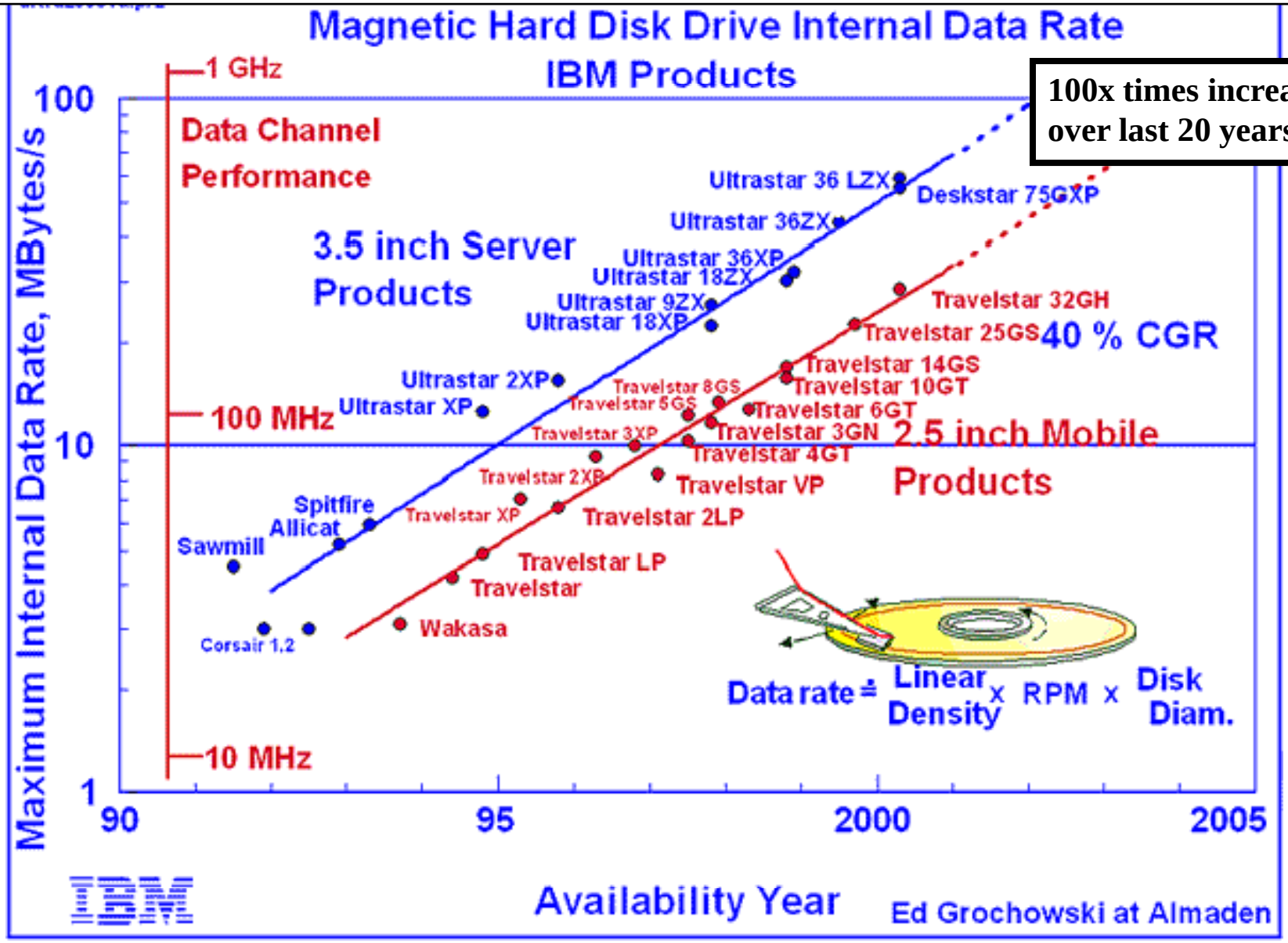
Historic Perspective of Hard Drive Characteristics Evolution: Areal Density



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.

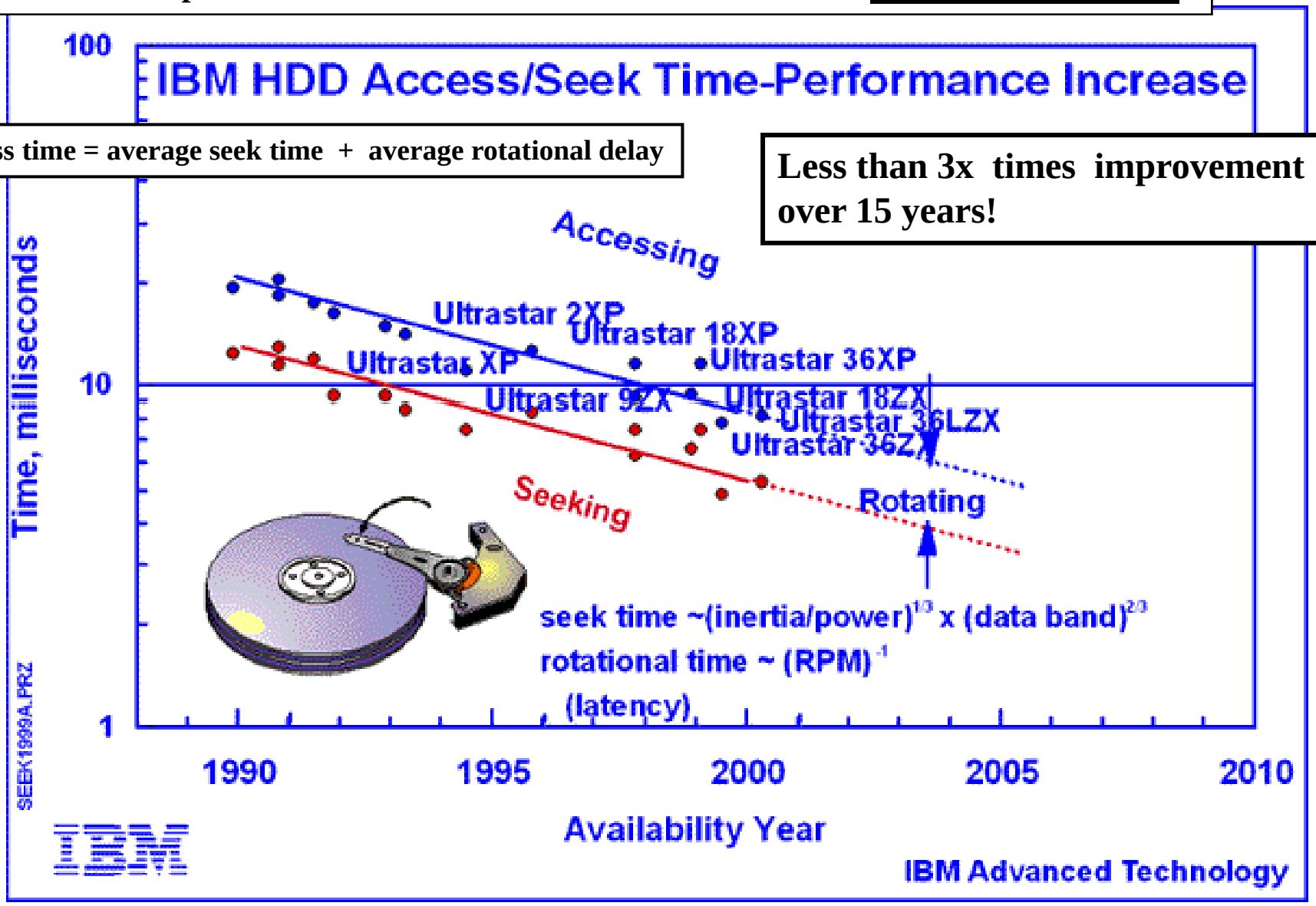
Current Areal Density ~ 500 Gbits / In^2

Historic Perspective of Hard Drive Characteristics Evolution: Internal Data Transfer Rate



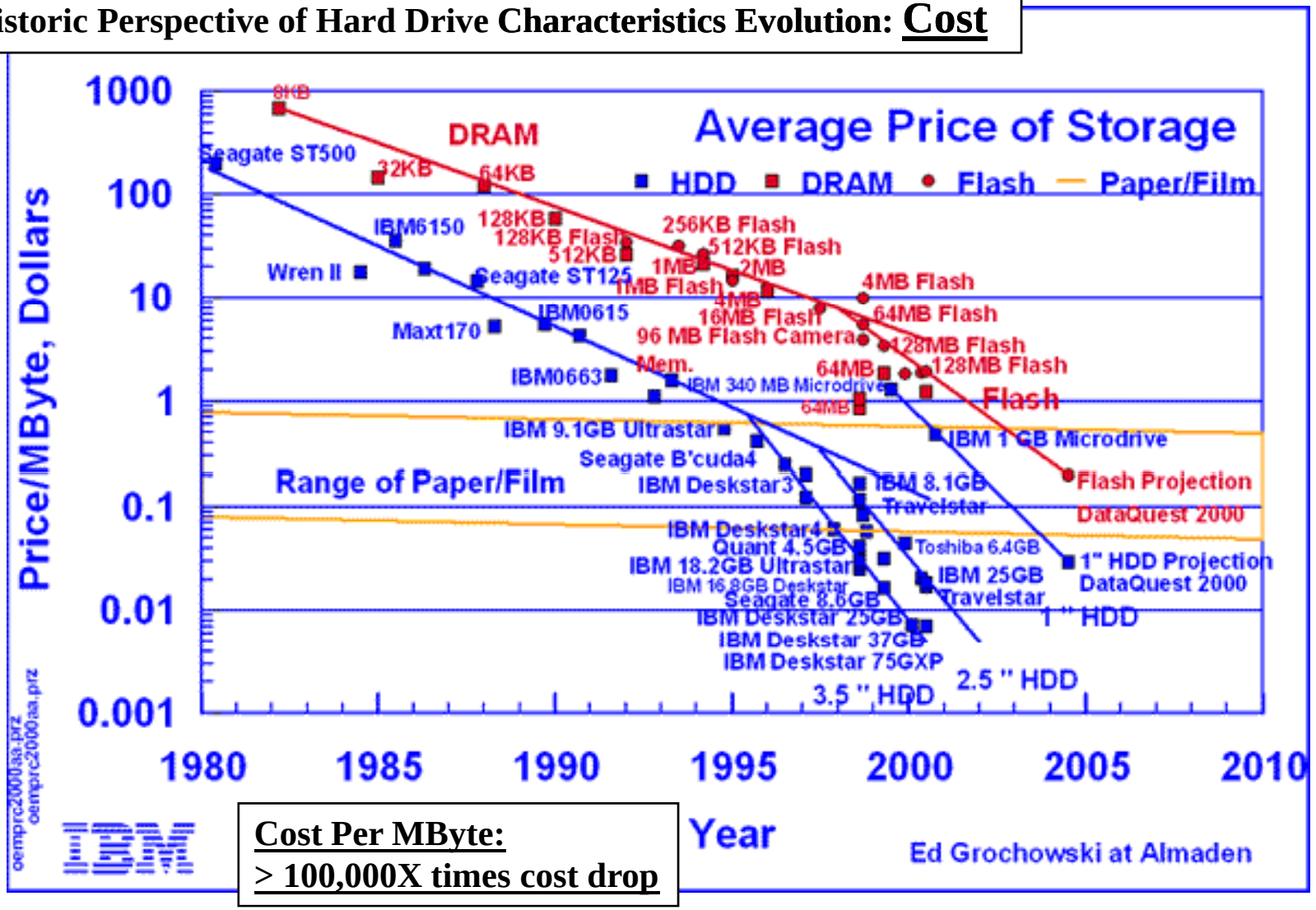
Internal data transfer rate increase is influenced by the increase in areal density

Historic Perspective of Hard Drive Characteristics Evolution: Access/Seek Time



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

Historic Perspective of Hard Drive Characteristics Evolution: Cost

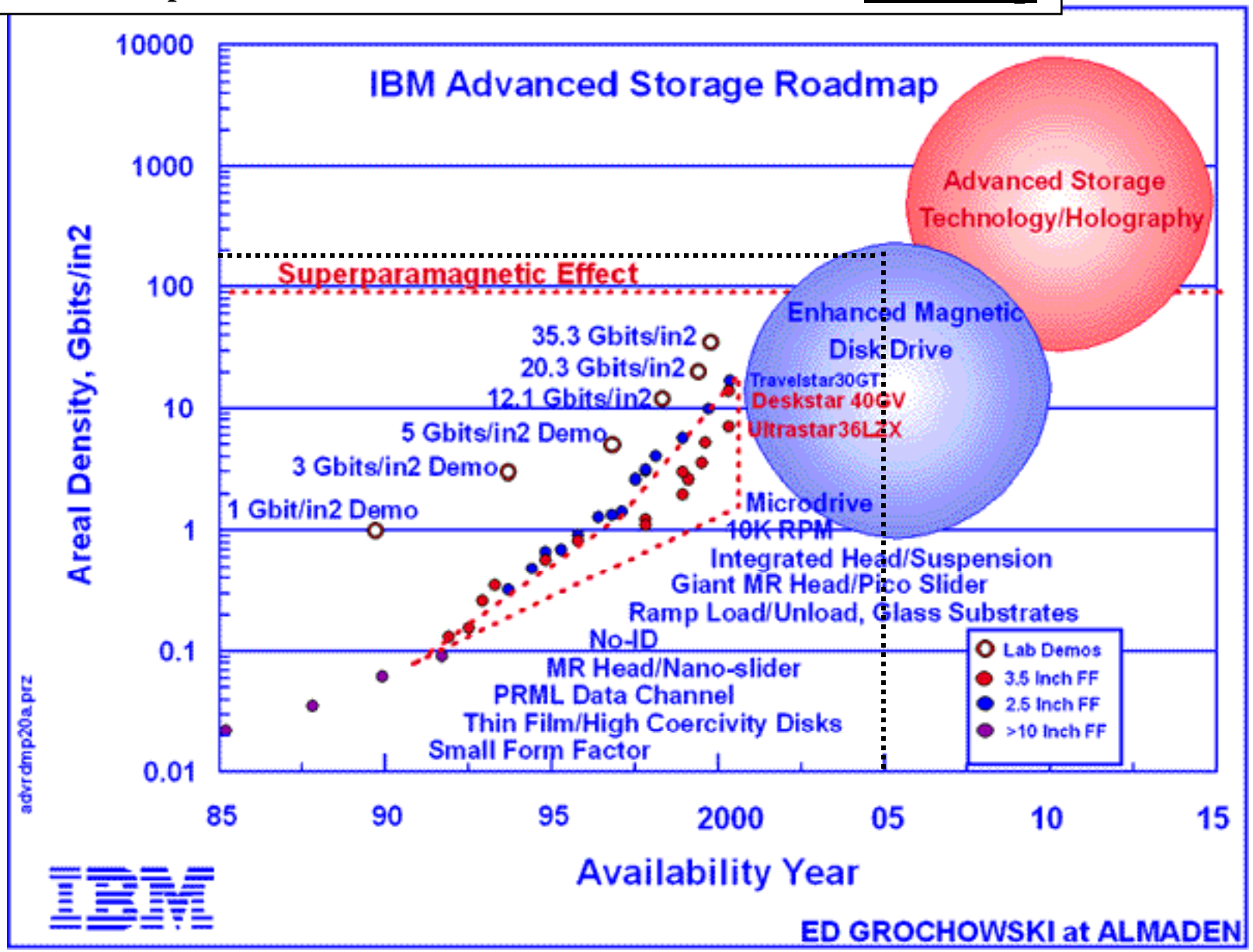


Cost Per MByte:
> 100,000X times cost drop

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.

Actual Current Hard Disk Storage Cost (First Quarter 2011):
~ 0.00005 dollars per MByte or about 20 GBytes /Dollar

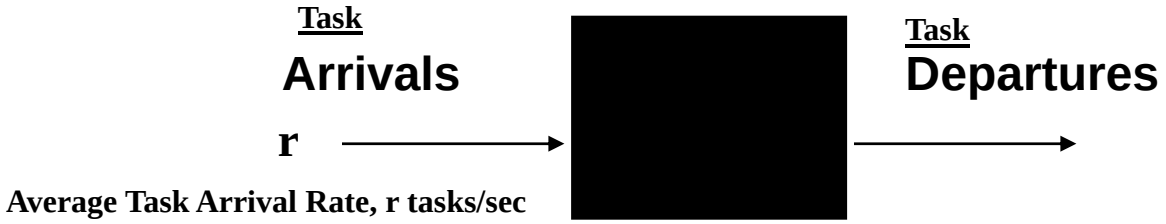
Historic Perspective of Hard Drive Characteristics Evolution: Roadmap



Current Areal Density ~ 500 Gbits / In^2

Introduction to Queuing Theory

(Steady State)



- **Concerned with long term, steady state than in startup:**
  - **where => Arrivals = Departures**
    Rate r Rate

- **Little's Law:** L_{sys} (length or number of tasks in system)

  Mean number tasks in system r (arrival rate) T_{sys} (System Time)

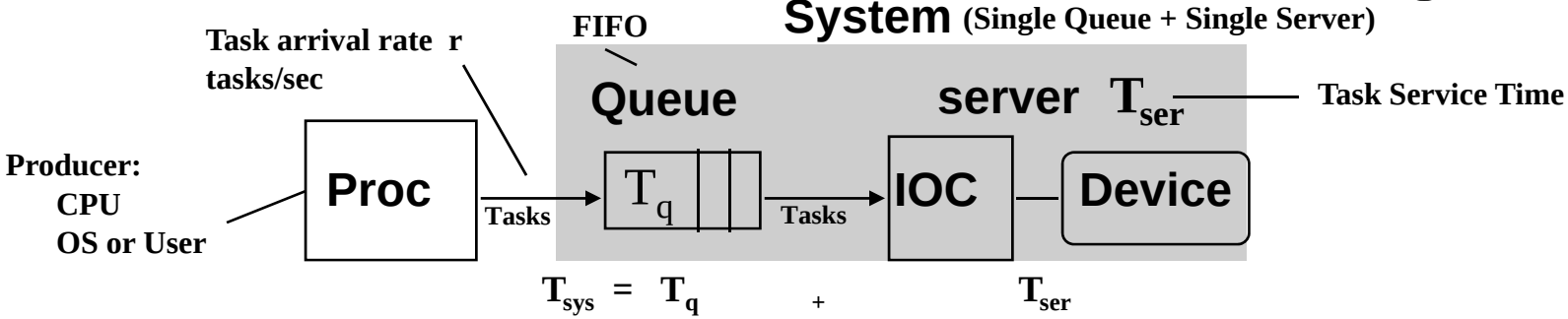
  i.e. average

  = arrival rate x mean response time

(Steady State)

- **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 thus $T_{sys} = T_{ser} + T_q$

  Ignoring CPU processing time and other system delays
  - 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)

AKA Loading Factor

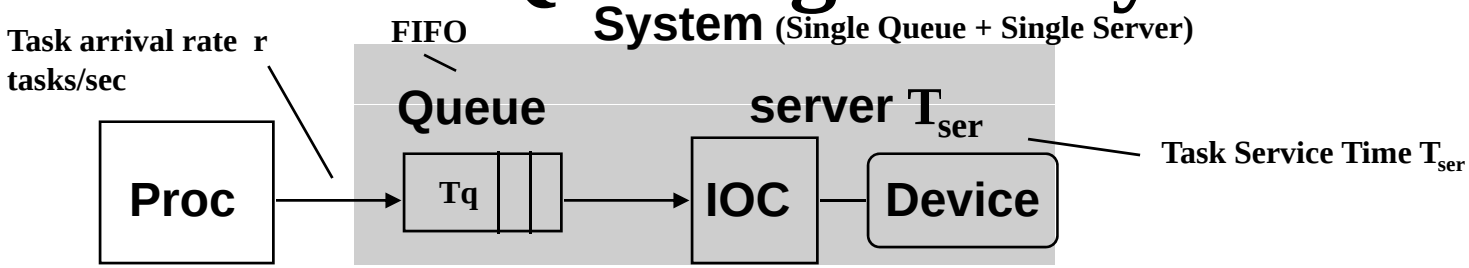
- **Server utilization = u = r / Service rate = r x T_{ser}**

r = Task Arrival rate

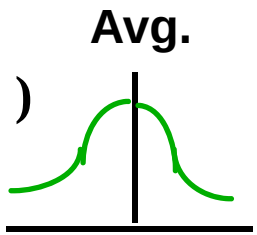
u must be between 0 and 1 otherwise there would be more tasks arriving than could be serviced

Here a server is the device (i.e hard drive) and its I/O controller (IOC)

A Little Queuing Theory



- **Server spends a variable amount of time with customers**
  - Arithmetic mean time = *m1* = (f1 x T1 + f2 x T2 +...+ fn x Tn)
    - **where T_i is the time for task i and f_i is the frequency of task i**
  - ***variance =*** $(f1 \times T1^2 + f2 \times T2^2 + ... + fn \times Tn^2) - m1^2$
    - **Must keep track of unit of measure (100 ms^2 vs. 0.1 s^2)**
  - ***Squared coefficient of variance*: $C^2 = variance/m1^2$**
    - **Unitless measure**

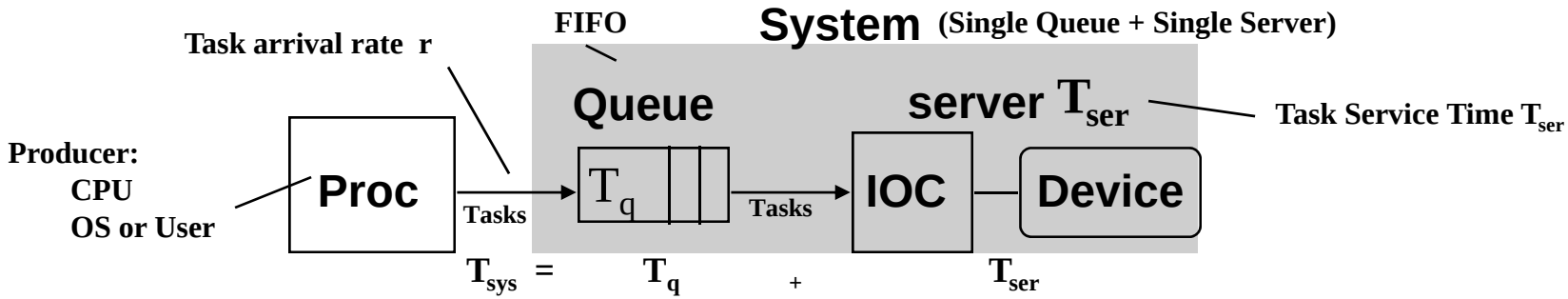


Distributions:

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

Variance = (Standard deviation)^2

A Little Queuing Theory



- **Service time completions vs. waiting time for a busy server: randomly <u>arriving task joins a queue</u> of arbitrary length when server is busy, otherwise <u>serviced immediately</u>**
  - **<u>Unlimited length queues key simplification</u>**
- **<u>A *single server queue*</u>: 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, $Time_{server}$**

<u>Response Time</u>

$$Time_{system} = Time_{queue} + Time_{server} = T_{sys} = T_q + T_{ser}$$

Ignoring CPU processing time and other system delays

$$Time_{queue} = Length_{queue} \times Time_{server} + \text{Time for the server to complete current task}$$

Time for the server to complete current task = Server utilization x remaining service time of current task

$$Length_{queue} = \text{Arrival Rate} \times Time_{queue} \quad \text{(Little's Law)}$$

We need to estimate waiting time in queue (i.e $Time_{queue} = T_q$)?

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

For Single Queue + Single Server

- **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}

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

$\text{Time}_{\text{queue}}$ = Average residual service time + $\text{Length}_{\text{queue}} \times \text{Time}_{\text{server}}$

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

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

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

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

(Rearrange)

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

$L_q = r \times T_q$ (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

Single Queue + Single Server

- **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}

Arrival Distribution (i.e $C^2 = 1$)

Service Distribution (i.e $C^2 = 1$)

Number of Servers

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

  (T_q average time/task in queue)

  T_{ser} average time to service a task

  L_q Average length of queue $L_q = r \times T_q = u^2 / (1 - u)$

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

Queuing Time, T_q

Response Time

$$Time_{system} = Time_{queue} + Time_{server} = T_{sys} = T_q + T_{ser}$$

(In Textbook page 726)

Single Queue + Multiple Servers (Disks/Controllers)

I/O Modeling: M/M/m Queue

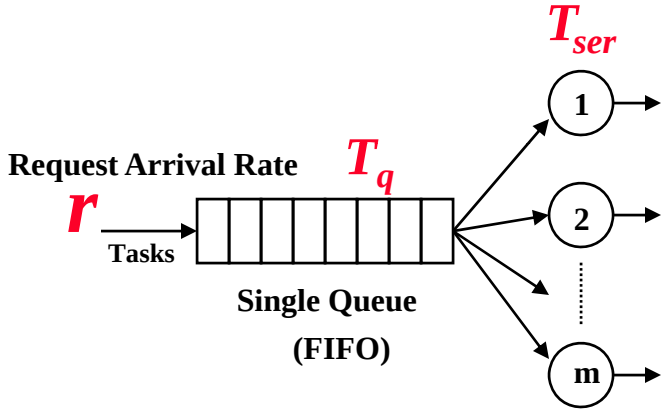
Arrival — Service — Number of servers

- I/O system with Markovian request arrival rate r (i.e $C^2 = 1$)
- A single queue serviced by m servers (disks + controllers) each with Markovian Service rate = $1/T_{ser}$ (i.e $C^2 = 1$)

(and requests are distributed evenly among all servers)

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

$$\text{where } u = r \times T_{ser} / m$$



m servers each has service time = Tser

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}?

r ; i.e $C^2 = 1$; T_{ser}

- We have:

  r average number of arriving requests/second = 40

  T_{ser} average time to service a request = 20 ms (0.02s)

- We obtain:

Utilization U

u server utilization: $u = r \times T_{ser}$ = 40/s x .02s = 0.8 or 80%

T_q average time/request in queue = $T_{ser} \times u / (1 - u)$
= 20 x 0.8/(1-0.8) = 20 x 0.8/0.2 = 20 x 4 = 80 ms (0 .08s)

i.e Mean Response Time → 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 x 0.08s = 3.2 requests in queue

L_{sys} average # tasks in system: $L_{sys} = r \times T_{sys}$ = 40/s x 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

i.e $C^2 = 1$

T_{ser}

(Changed from 20 ms to 10 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} ?
- **We have:**

  r average number of arriving requests/second = 40

  T_{ser} average time to service a request = 10 ms (0.01s)

- **We obtain:**

  Utilization U

  u server utilization: $u = r \times T_{ser}$ = 40/s x .01s = 0.4 or 40%

  T_q average time/request in queue = $T_{ser} \times u / (1 - u)$
  = 10 x 0.4/(1-0.4) = 10 x 0.4/0.6 = 6.67 ms (0 .0067s)

  i.e Mean Response Time → T_{sys} 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

CPU

- **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).

I/O

Service Time, Tser, Throughput

Tq

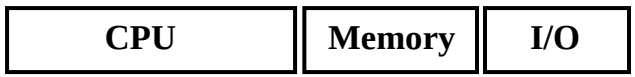
- **Memory subsystem performance:**
  - Available memory bandwidth for I/O operations (For DMA)

Memory

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

OS

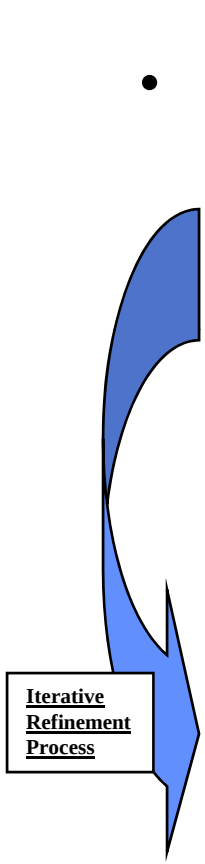
Components of Total System Execution Time:

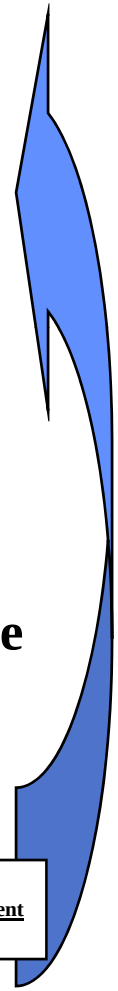


System performance depends on many aspects of the system
("limited by weakest link in the chain"): **The system performance bottleneck**

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:** (i.e system configurations)
    - **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") → System Performance Bottleneck

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** — Main system 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** — *(i.e u = 1)*
  - **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.** — *(i.e u = 1)*

    (Since I/O queuing delays are ignored here 100% disk utilization is allowed)
- **What is the average IOPS?** — i.e I/O throughput
- **What is the average I/O bandwidth?**
- **What is the average response time per IO operation?**

Here: 1KBytes = 10^3 bytes, MByte = 10^6 bytes, 1 GByte = 10^9 bytes

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):**

Determining the system performance bottleneck

  - **CPU :** (500 MIPS)/(10,000 instructions per I/O) = 50,000 IOPS
    CPU time per I/O = 10,000 / 500,000,000 = .02 ms
  - **Main Memory :** (16 bytes)/(100 ns x 16 KB per I/O) = 10,000 IOPS
    Memory time per I/O = 1/10,000 = .1ms
  - **I/O bus:** (200 MB/sec)/(16 KB per I/O) = 12,500 IOPS
  - **SCSI-2:** (20 buses)/((1 ms + (16 KB)/(20 MB/sec)) per I/O) = 11,111 IOPS
    SCSI bus time per I/O = 1ms + 16/20 ms = 1.8ms
  - Disks: (100 disks)/((8 ms + 0.5/(7200 RPMS) + (16 KB)/(6 MB/sec)) per I/O) = 6700 IOPS

T_{ser}

T_{disk} = (8 ms + 0.5/(7200 RPMS) + (16 KB)/(6 MB/sec) = 8+ 4.2+ 2.7 = 14.9ms

Throughput:

- **The disks limit the I/O performance to 6700 IOPS**
- **The average I/O bandwidth is 6700 IOPS x (16 KB/sec) = 107.2 MB/sec**
- ***Response Time Per I/O = Tcpu + Tmemory + Tscsi + Tdisk =***
  = .02 + .1 + 1.8 + 14.9 = 16.82 ms

Since I/O queuing delays are ignored here 100% disk utilization is allowed

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)

Here m = 100

- Assume the following system components:
  - 500 MIPS CPU
  - 16-byte wide memory system with 100 ns cycle time
  - 200 MB/sec I/O bus — Main system 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). — Thus maximum utilization of any system component is fixed in question at u = 0.6 or 60%
  - 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? — i.e I/O throughput
- Average response time per IO operation?

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):**

  Determining the system performance bottleneck

  - **CPU : (500 MIPS)/(10,000 instr. per I/O) x .6 = 30,000 IOPS**
    CPU time per I/O = 10,000 / 500,000,000 = .02 ms
  - **Main Memory : (16 bytes)/(100 ns x 16 KB per I/O) x .6 = 6,000 IOPS**
    Memory time per I/O = 1/10,000 = .1ms
  - **I/O bus: (200 MB/sec)/(16 KB per I/O) x .6 = 12,500 IOPS**
  - **SCSI-2: (20 buses)/((1 ms + (16 KB)/(20 MB/sec)) per I/O) x .6 = 6,666.6 IOPS**
    SCSI bus time per I/O = 1ms + 16/20 ms = 1.8ms
  - **Disks: (100 disks)/((8 ms + 0.5/(7200 RPMS) + (16 KB)/(6 MB/sec)) per I/O) x .6 = 6,700 x .6 = 4020 IOPS**

  T_{ser} = (8 ms + 0.5/(7200 RPMS) + (16 KB)/(6 MB/sec) = 8+4.2+2.7 = 14.9ms

- **The disks limit the I/O performance to r = 4020 IOPS**

  Throughput

- **The average I/O bandwidth is 4020 IOPS x (16 KB/sec) = 64.3 MB/sec**
- ***$T_q = T_{ser} \times u / [m(1-u)] = 14.9ms \times .6 / [100 \times .4] = .22$ ms***

  Using expression for Tq for M/M/m from slide 36

- ***Response Time = Tser + Tq + Tcpu + Tmemory + Tscsi =***

  14.9 + .22 + .02 + .1 + 1.8 = 17.04 ms

Total System response time including CPU time and other delays

Here: 1KBytes = 10^3 bytes, MByte = 10^6 bytes, 1 GByte = 10^9 bytes