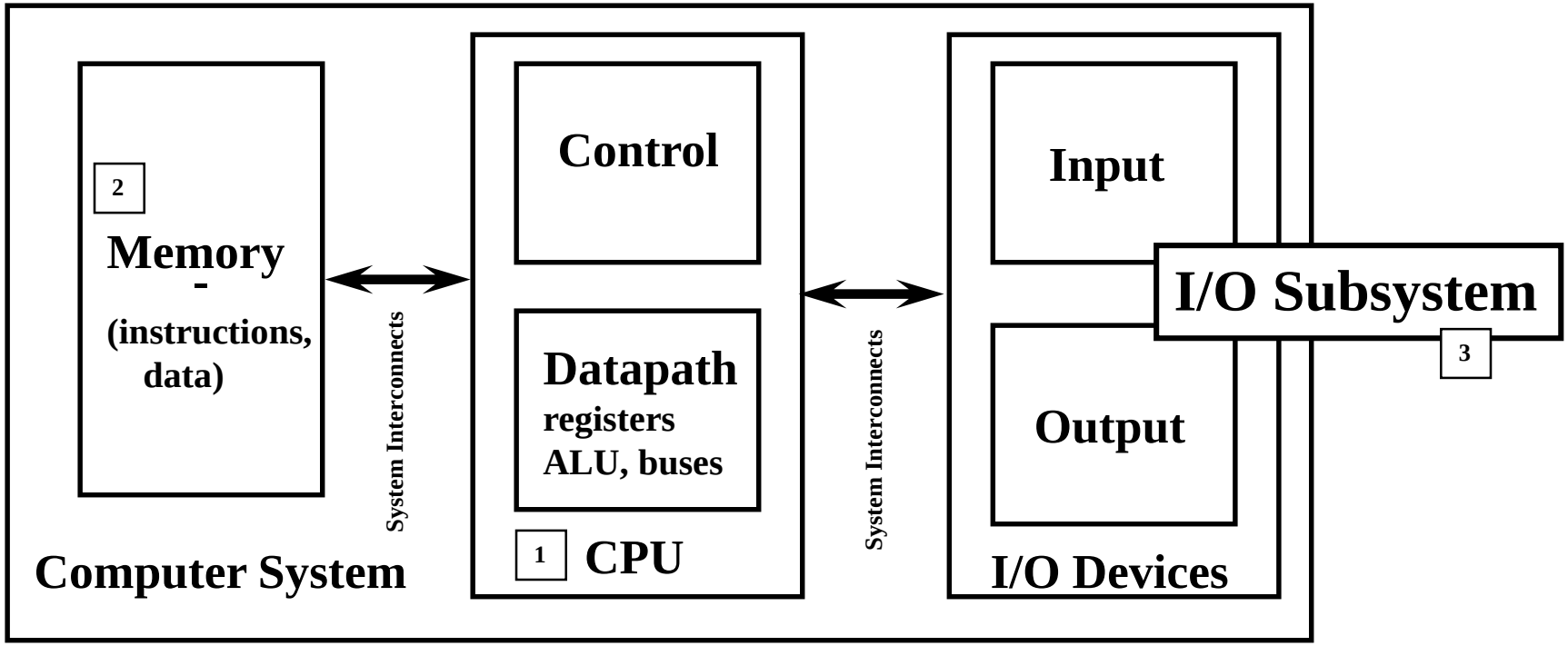
# I/O Subsystem

# 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



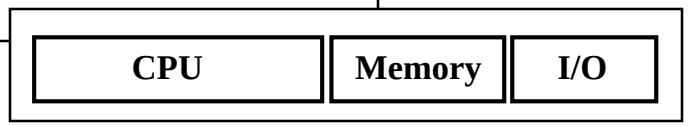

**System performance depends on many aspects of the system ("limited by weakest link in the chain"): The system performance bottleneck**

# 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).**
  - Including users
  - Including memory

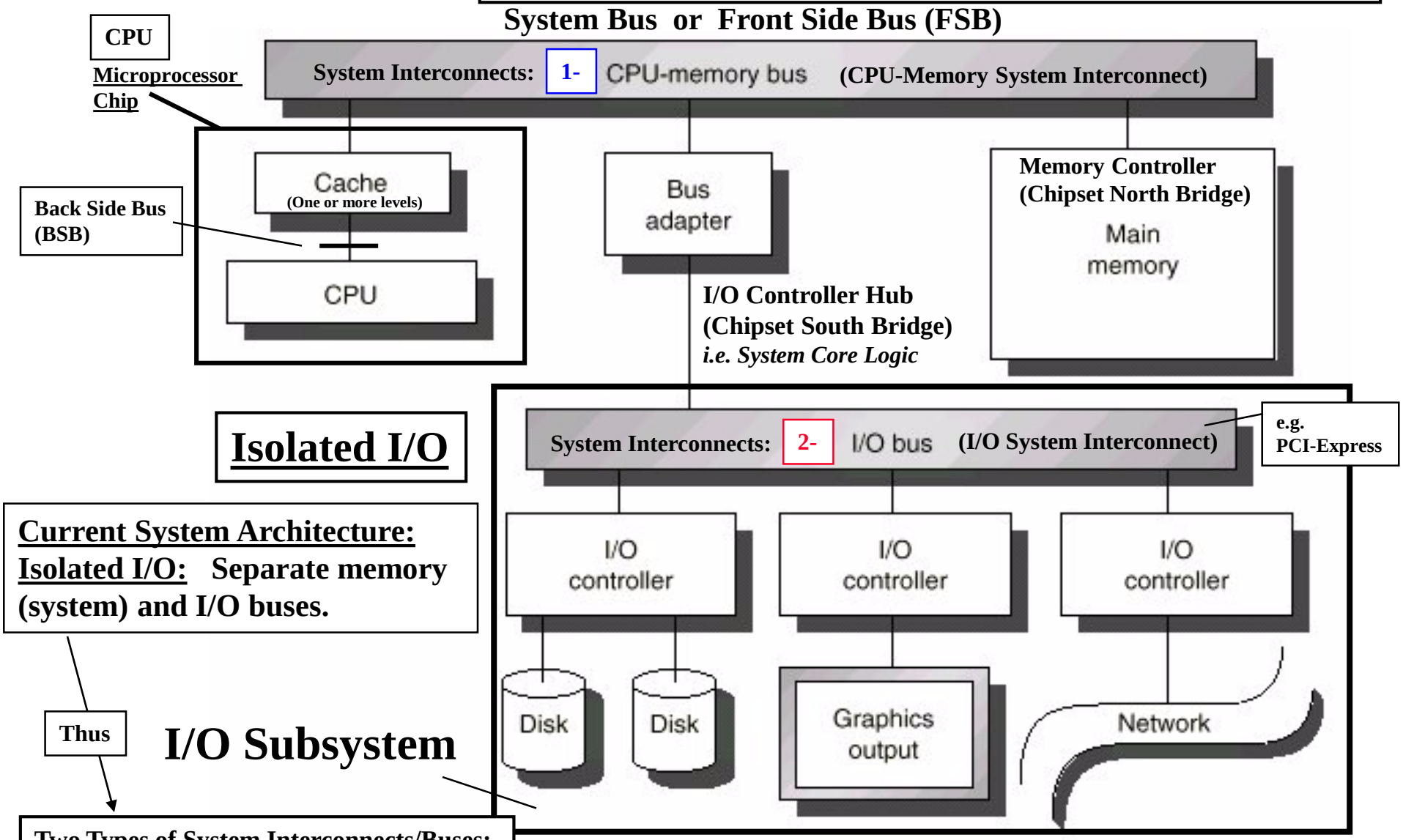
- **Design factors:**
  - **I/O device characteristics (input, output, storage, etc.) /Performance.**
  - **I/O Connection Structure (degree of separation from memory operations).** → Isolated I/O System Architecture
  - **I/O interface (the utilization of dedicated I/O and bus controllers).**
  - **Types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




**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**

# 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

CPU
L1
L2
Caches
L3
(possibly on-chip)
Front Side Bus (FSB)
System Bus

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

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

Memory Controller

Double Date Rate (DDR) SDRAM
PC3200
200 MHz DDR
64-128 bits wide
4-way interleaved
~3.2 GBYTES/SEC (64bit)

Memory Bus

Memory

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

Chipset
(System Logic)
North Bridge

Bus Adapter
e.g. PCI-Express

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 Controllers
NICs

Disks
Displays
Keyboards

Networks

Chipset
South Bridge
(System Logic)

I/O Devices

(Isolated I/O Subsystem)
I/O Subsystem

**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)

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

Dedicated Links

## 2 I/O buses/Interconnects:

Sometimes called I/O *channels or interfaces*

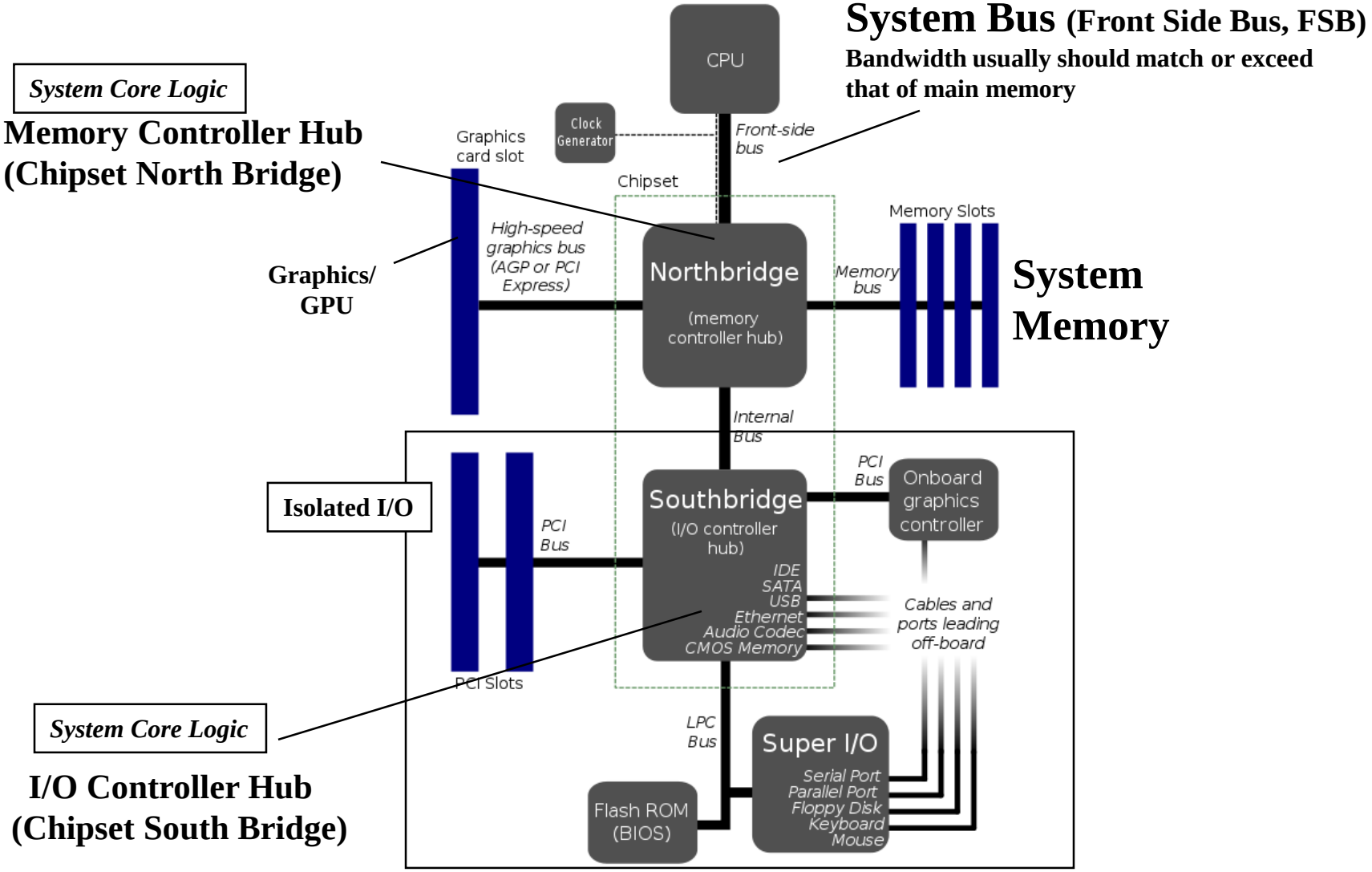
- Follow bus/interface industry standards.
- Usually formed by I/O interface adapters to handle many types of connected I/O devices.
- Wide range in the data bandwidth and latency
- Not usually interfaced directly to memory instead connected to processor-memory bus via a bus adapter (system chipset south 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

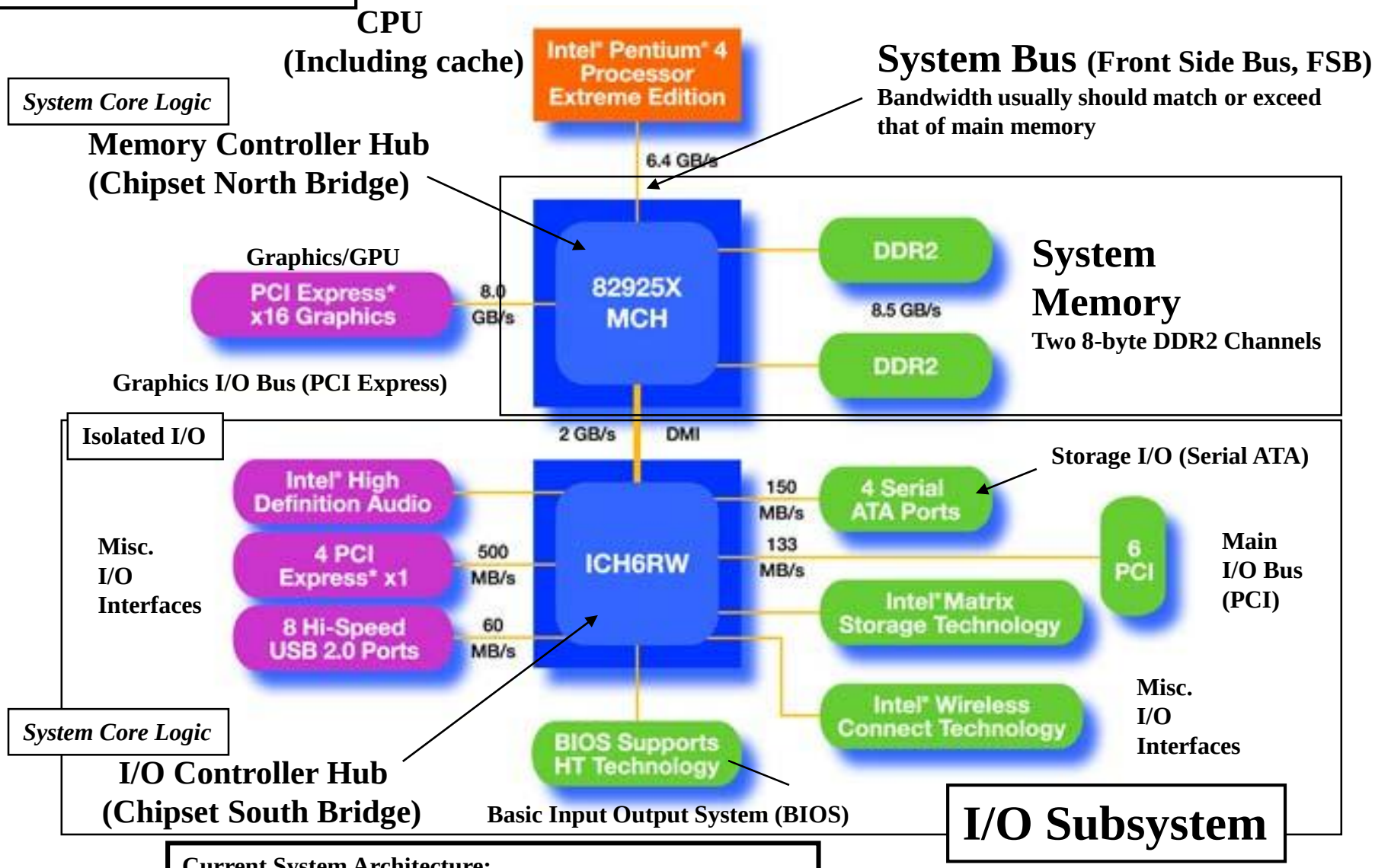


# Intel Pentium 4 System Architecture

## (Using The Intel 925 Chipset)

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

**And Core 2**

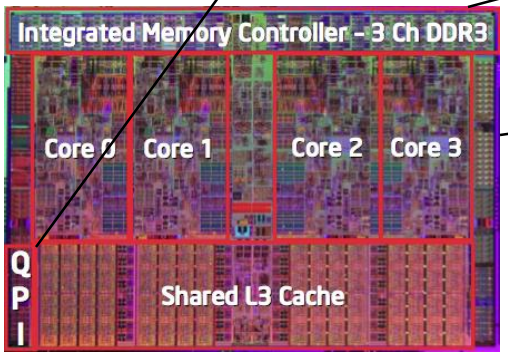



**Current System Architecture:**
**Isolated I/O: Separate memory and I/O buses.**

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

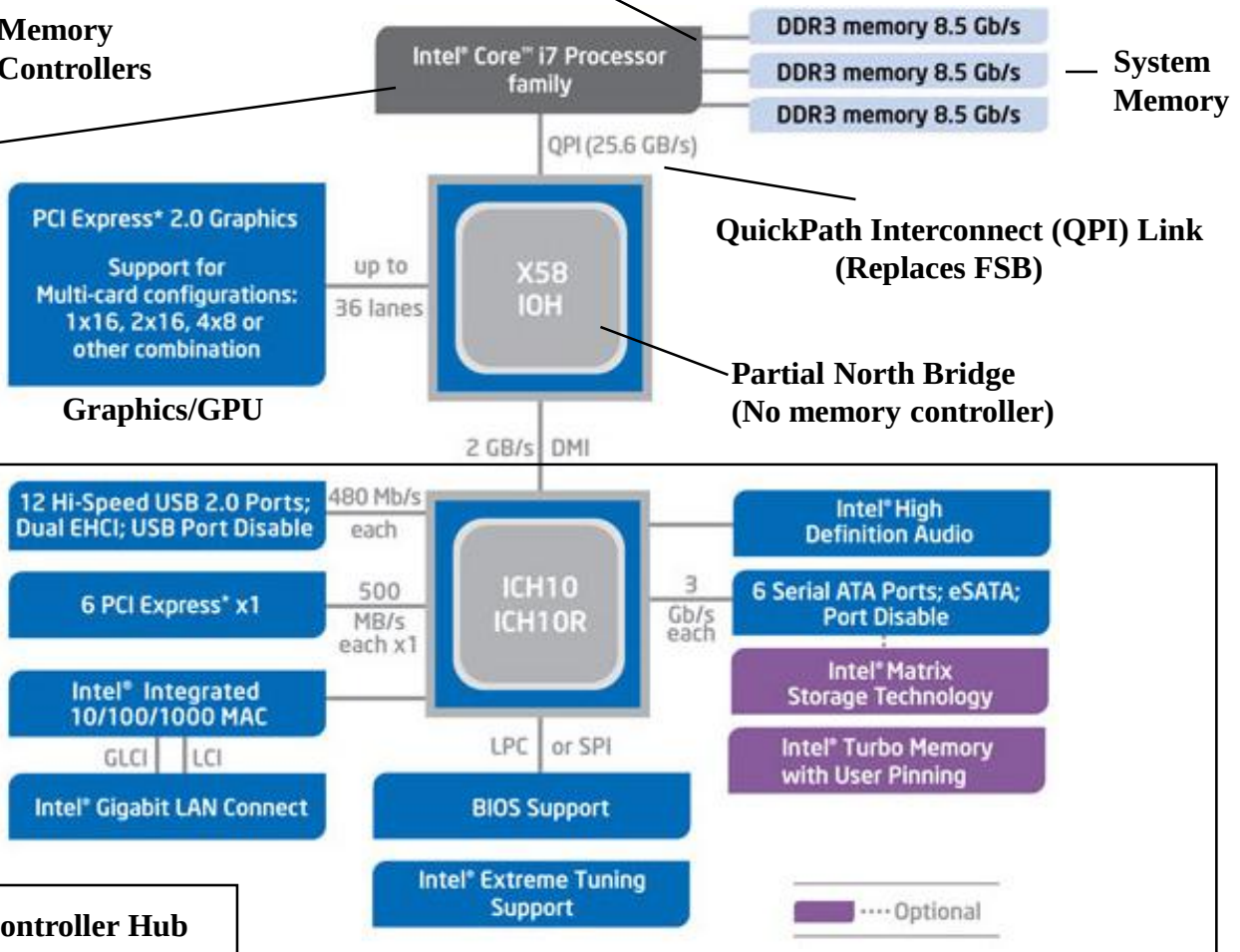
# Intel Core i7 “Nehalem” System Architecture

**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)**




**Memory Controllers**

**QPI Link(s)**




**System Memory**

**QuickPath Interconnect (QPI) Link (Replaces FSB)**

**Graphics/GPU**

**Partial North Bridge (No memory controller)**

**Isolated I/O**

**I/O Controller Hub (Chipset South Bridge)**

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

# Bus Characteristics

(e.g . FSB)

| *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 |
| | PCI 2.3 | 64 | 33.3 | 266 |
| | PCI 2.3 | 64 | 66.6 | 533 |
| | 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)<br>20MHz (Ultra)<br>40MHz (Ultra2)<br>80MHz (Ultra3)<br>160MHz (Ultra4) |
| Bus Masters | 1 | Multiple |
| Max no. devices | 2 | 7 (8-bit bus)<br>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.)

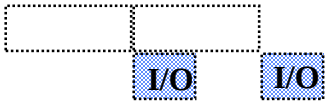
Memory-mapped register

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:**

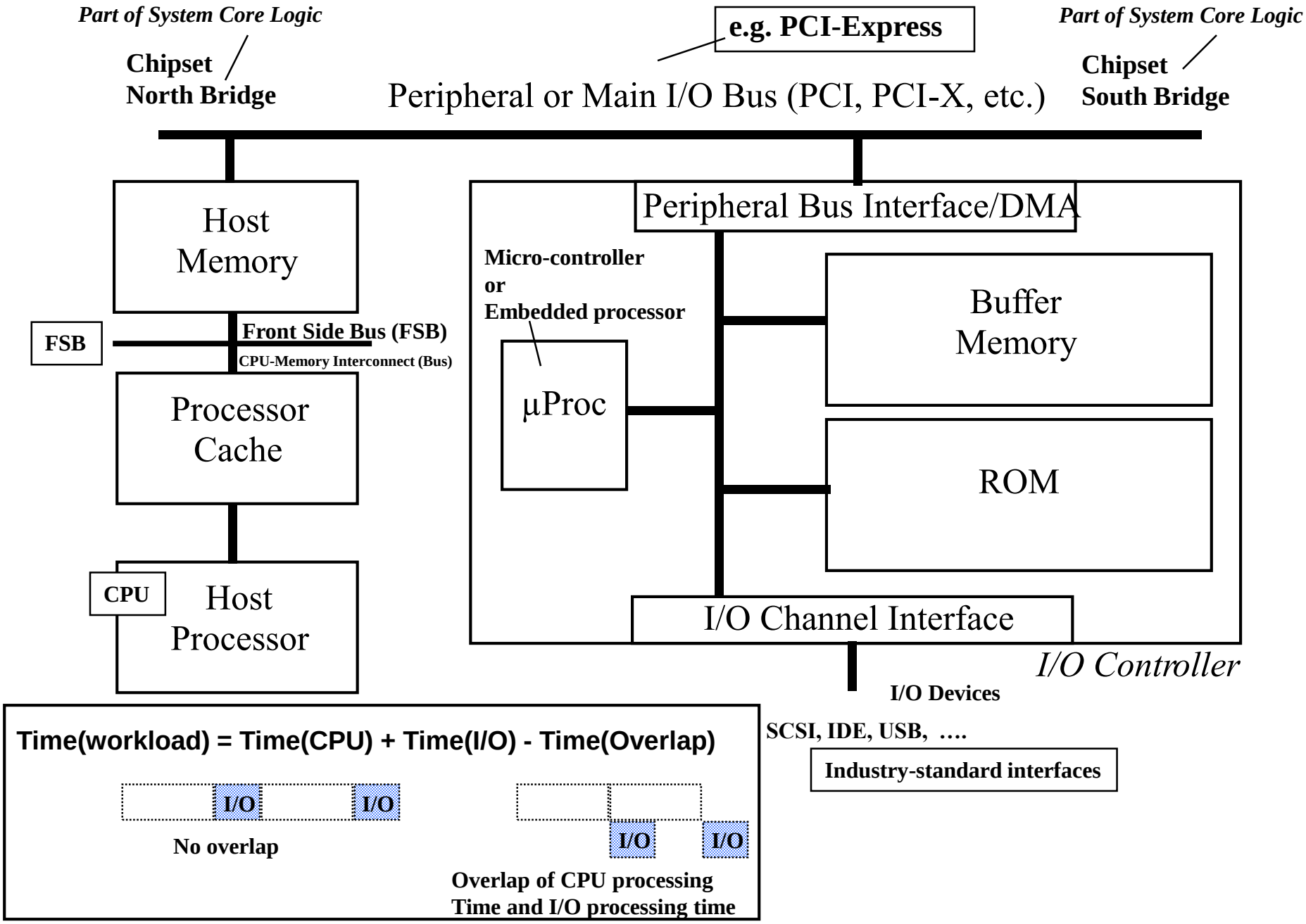
- **Specific to each type of I/O device/interface standard.**
- **To the CPU, and I/O device, it consists of a set of control and data registers (usually memory-mapped) within the I/O address space.**
- **On the I/O device side, it forms a localized I/O bus which can be shared by several I/O devices**
  - **(e.g IDE, SCSI, USB ...)** Industry-standard interfaces

Why?

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

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

# 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).**

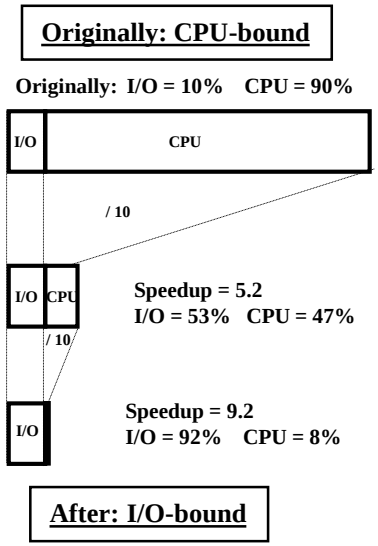
i.e storage devices (hard drives)


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

To Start With →

**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)**



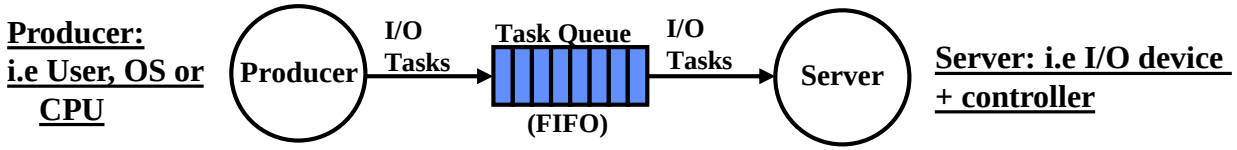

- **The I/O system performance bottleneck diminishes the benefit of faster CPUs on overall system performance.**

**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 task rate:**

  – **Measured in:** I/O Tasks/sec

    - **Accesses/second or,**
    - **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.**

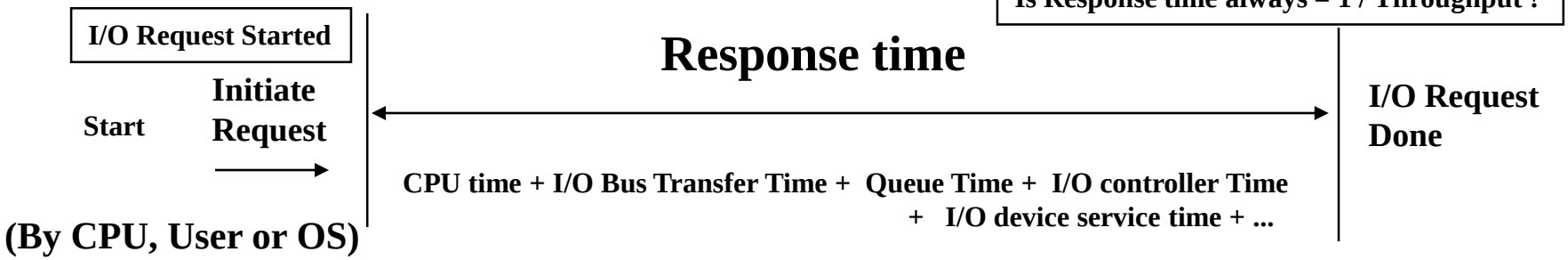
# System & I/O Performance Metrics: Response time

- **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 ?**

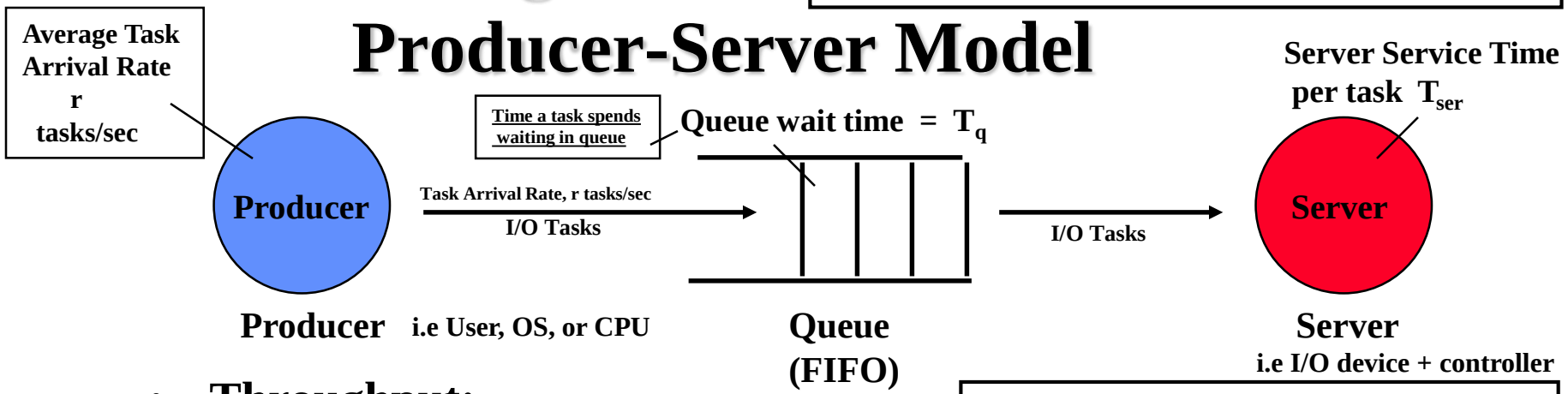



**The utilization of DMA and I/O device queues and multiple I/O devices servicing a queue may make throughput >> 1 / response time**

# 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:

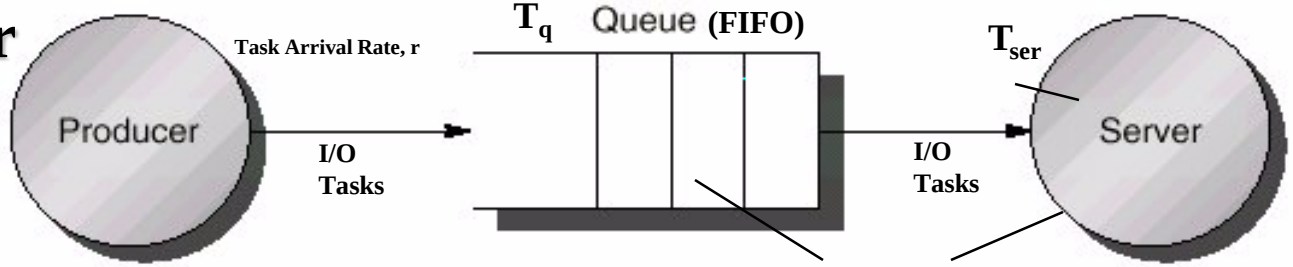
- The queue should be empty (no waiting time in queue).
- The server will be idle at times.

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

# Producer-Server Model

Single Queue + Single Server

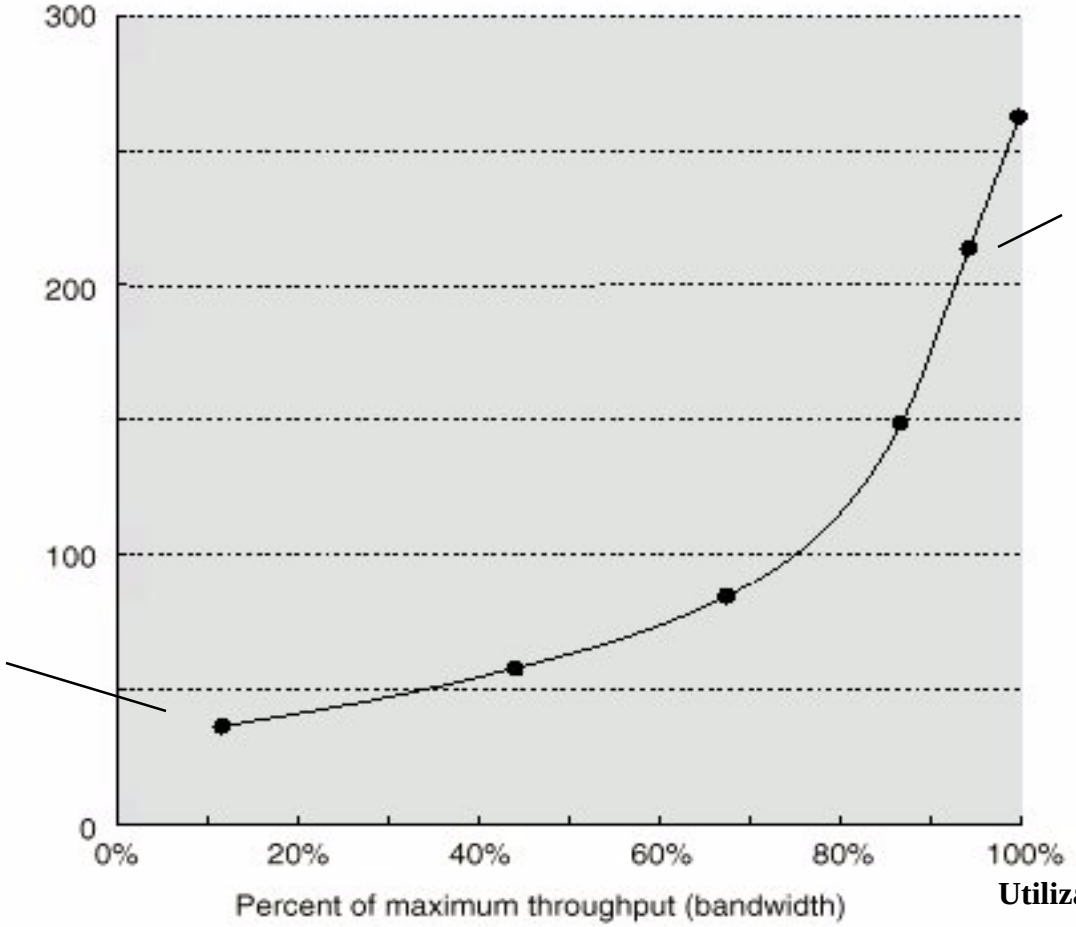
User or CPU




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

I/O device + controller

# Throughput vs. Response Time




Queue full most of the time. More time in queue

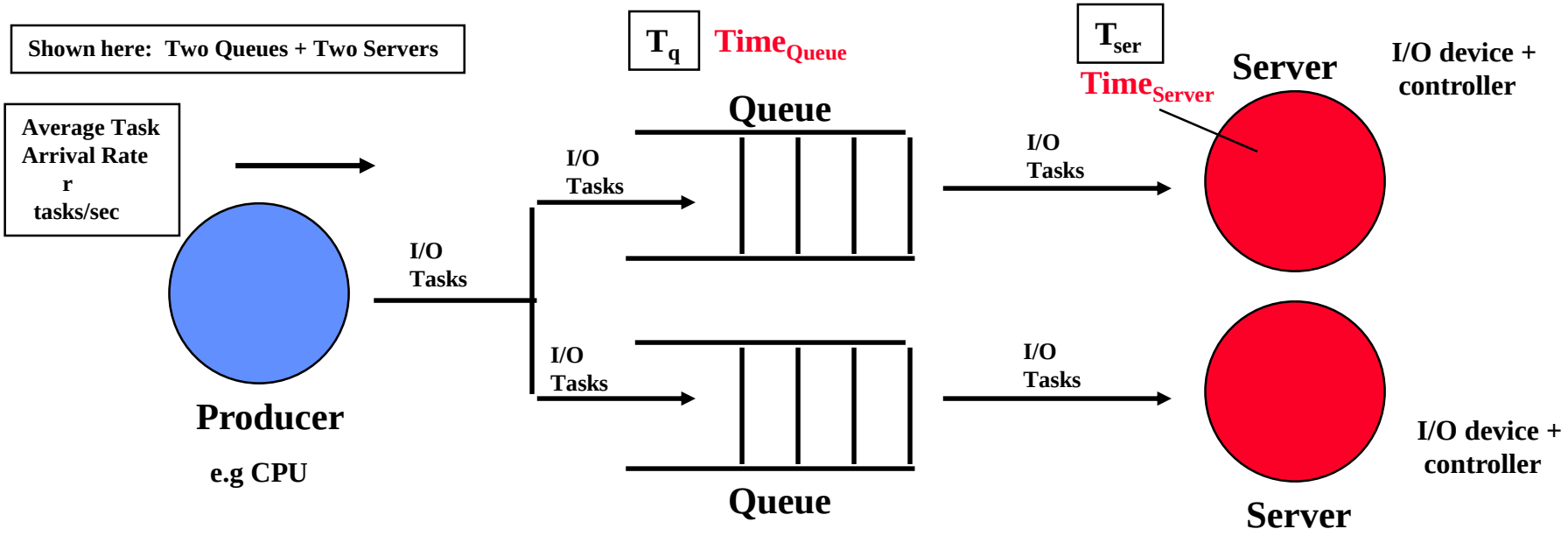
Queue almost empty most of the time
Less time in queue

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 = $Time_{System}$ = $Time_{Queue}$ + $Time_{Server}$ = $T_q$ + $T_{ser}$

Storage I/O Systems:

i.e. Rotation-Based Storage: Hard Drives

# Magnetic Disks

## Characteristics:

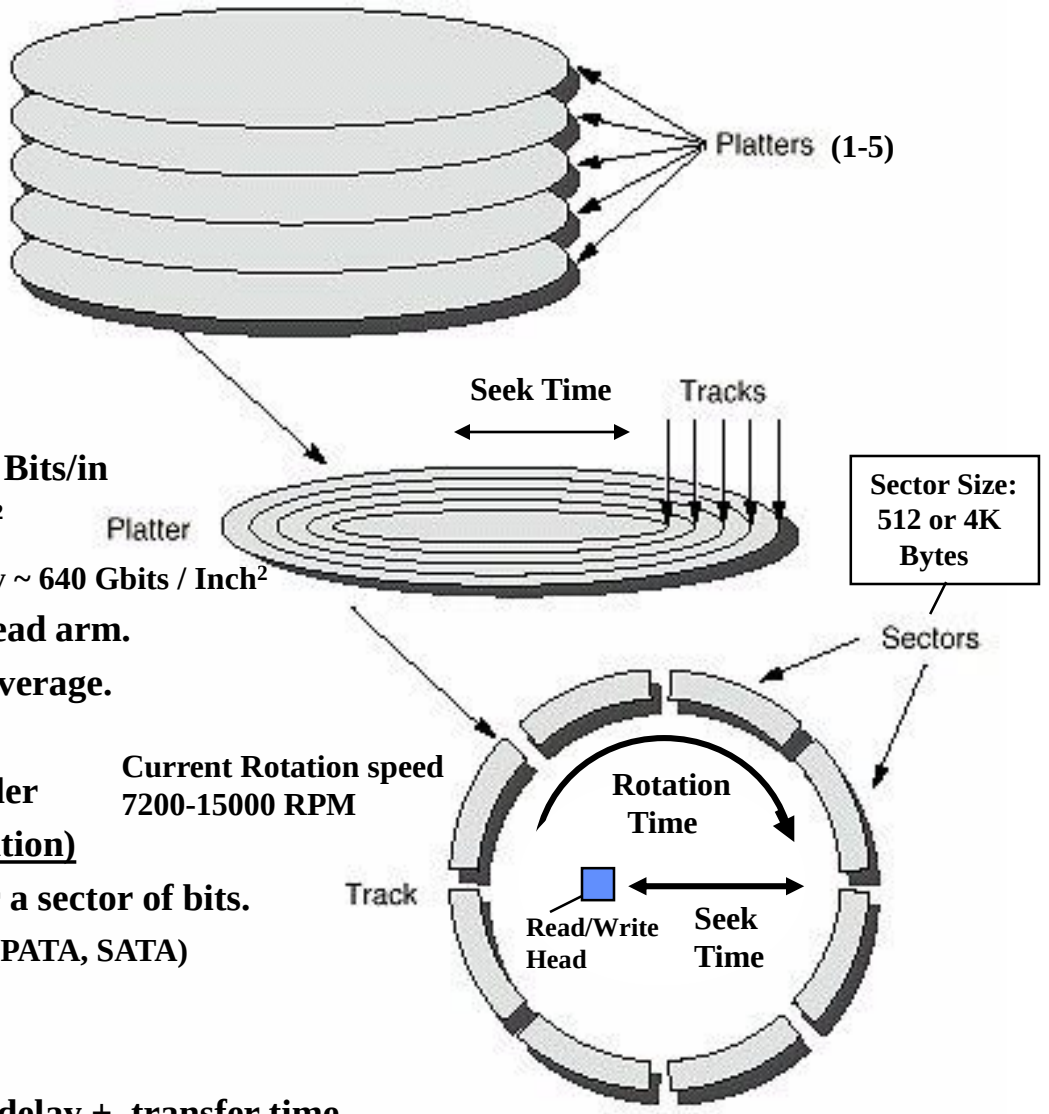
- Diameter (form factor): 1.8in - 3.5in
- Rotational speed: 5,400 RPM-15,000 RPM
- Tracks per surface.
- Sectors per track: Outer tracks contain more sectors.
- Recording or Areal Density: Tracks/in X Bits/in
- Cost Per Megabyte.
- Seek Time: (2-12 ms)
  The time needed to move the read/write head arm.
  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

Bits/ Inch$^2$

Current Areal Density ~ 640 Gbits / Inch$^2$



# 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?

i.e. $T_{queue} = T_q = 0$

i.e. $T_{Ser}$

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

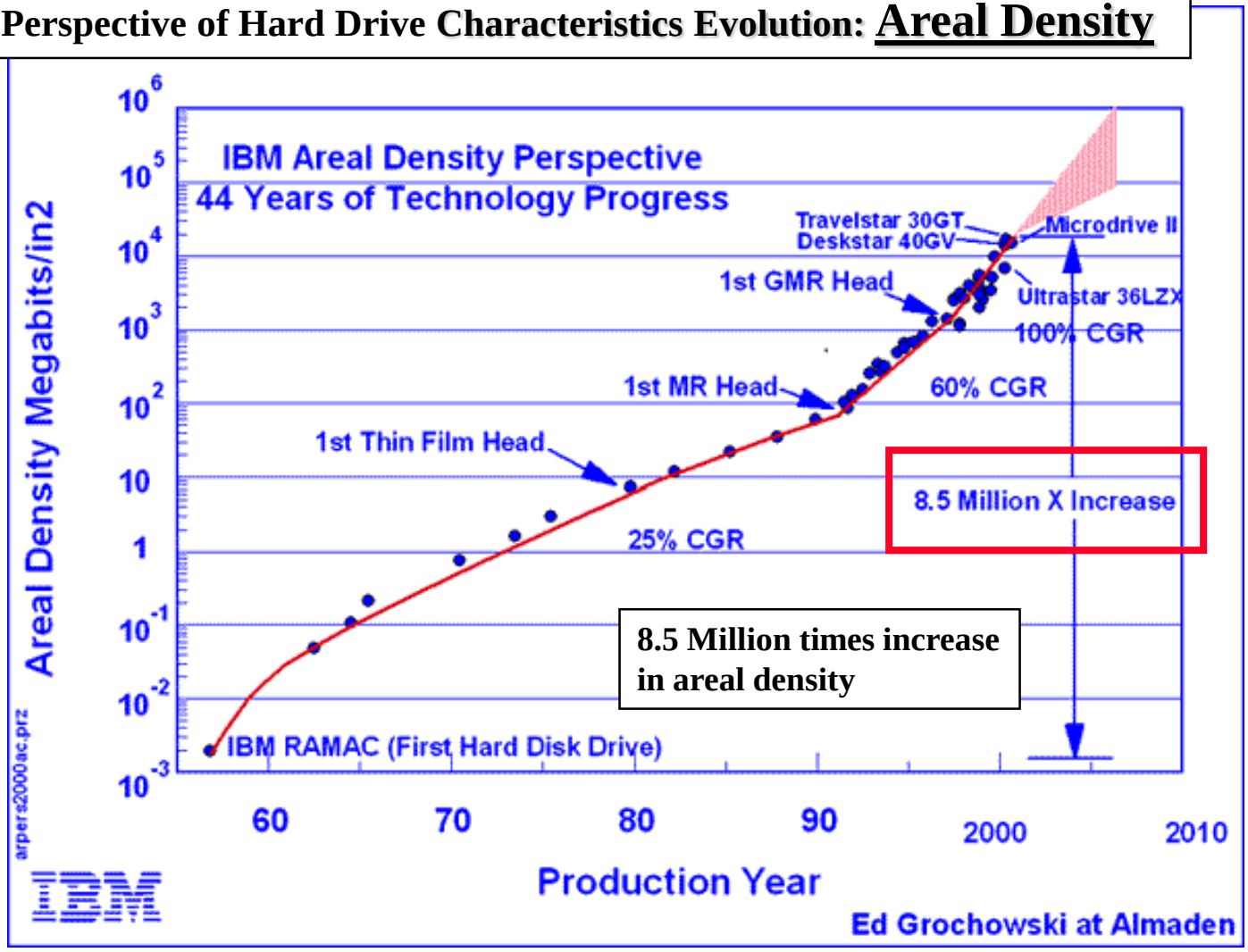
Actual time to process the disk request
is greater and may include CPU I/O processing Time
and queuing time

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

Or $T_{ser}$

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

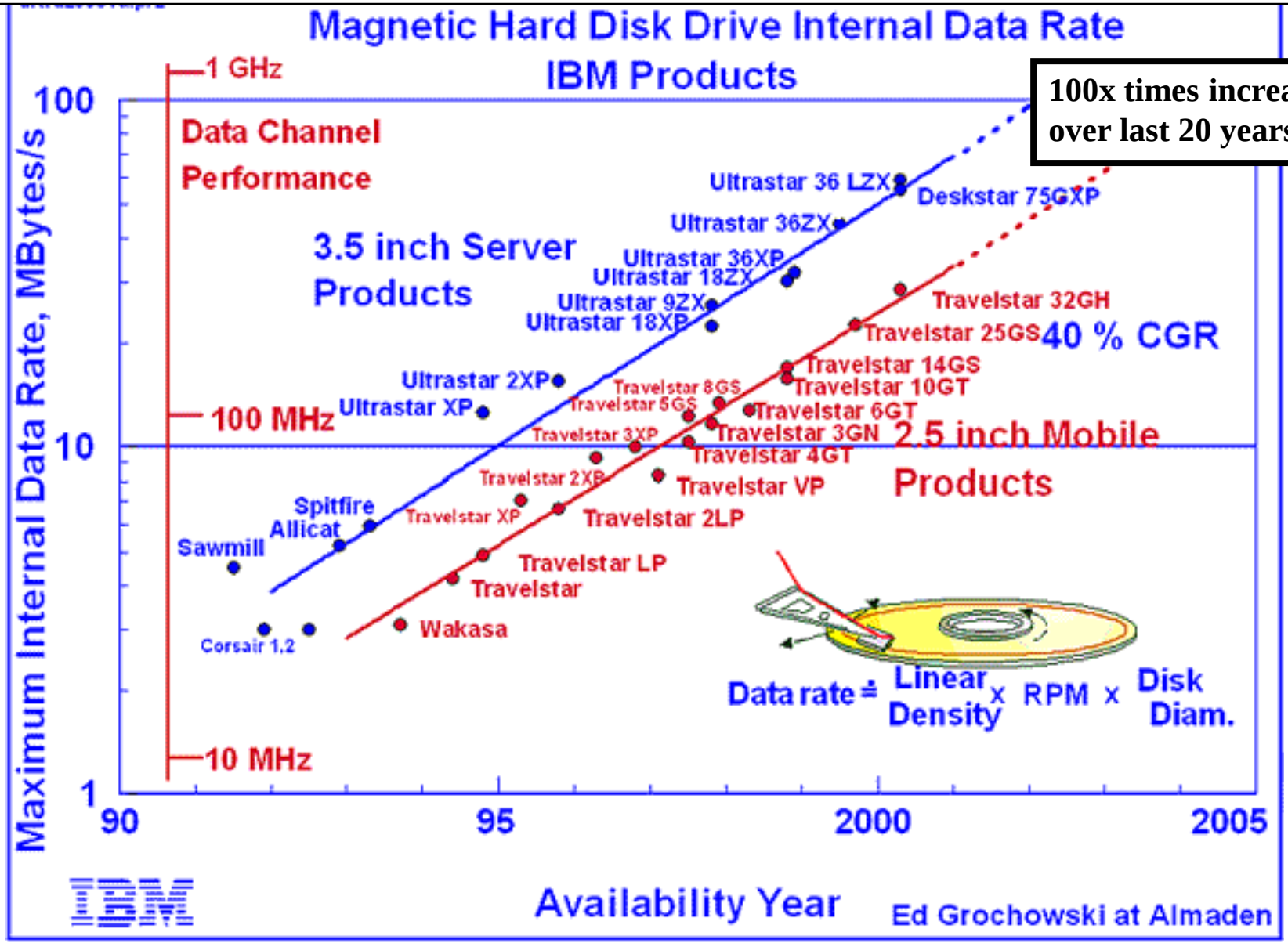
# Historic Perspective of Hard Drive Characteristics Evolution: Areal Density




**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.**

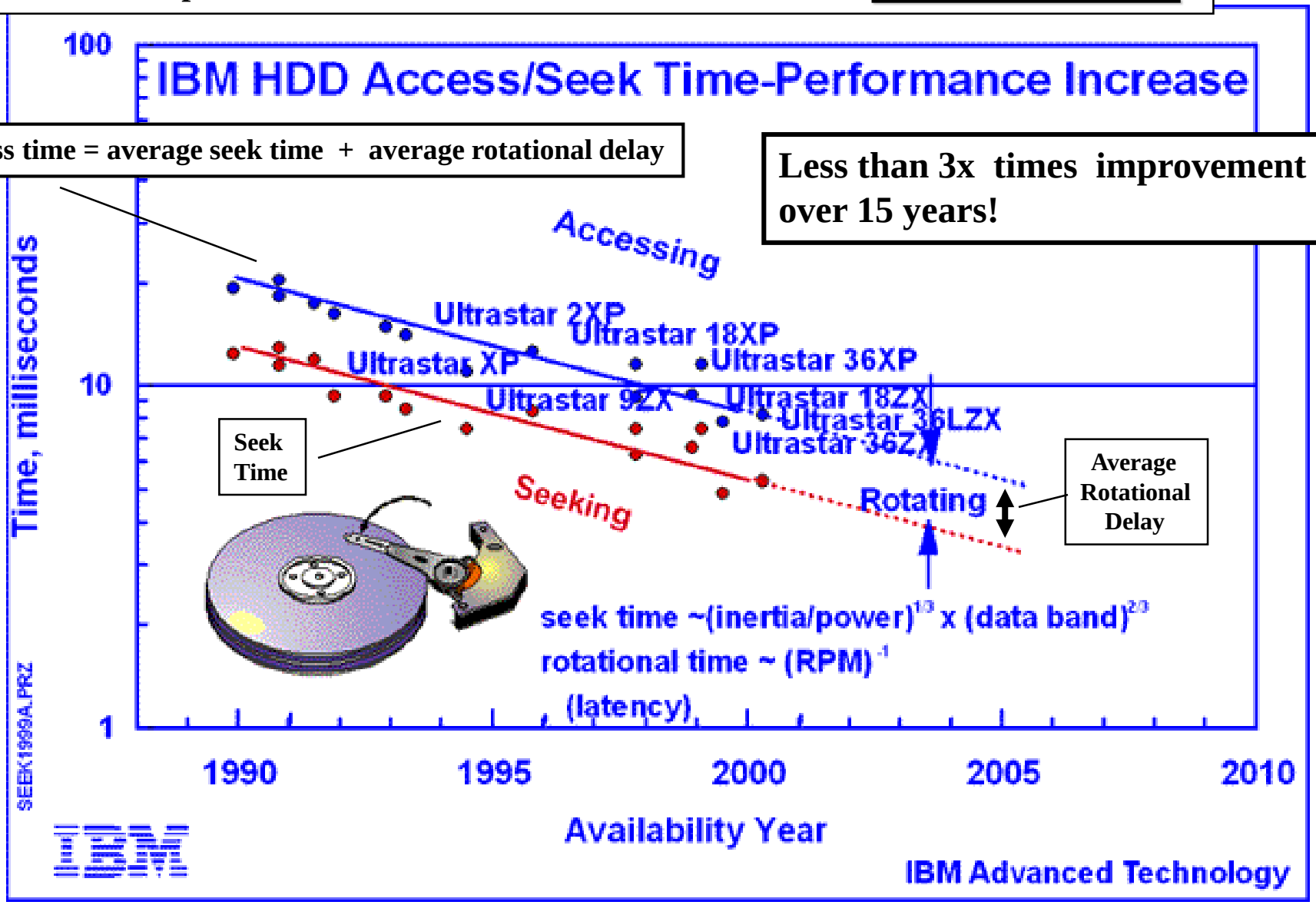
**Current Areal Density ~ 640 Gbits / In$^2$**

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



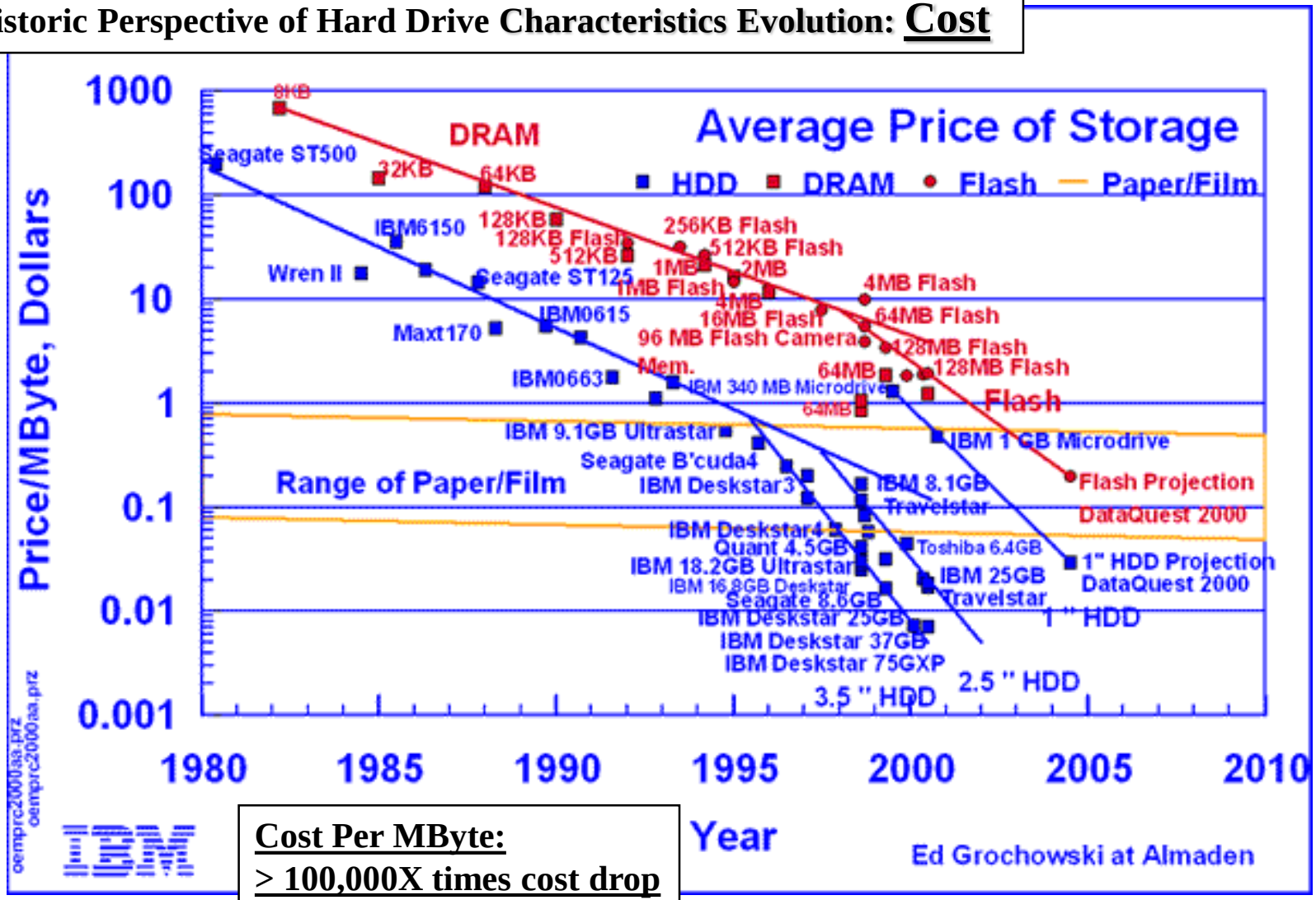

Internal data transfer rate increase is influenced by the increase in areal density

# 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**

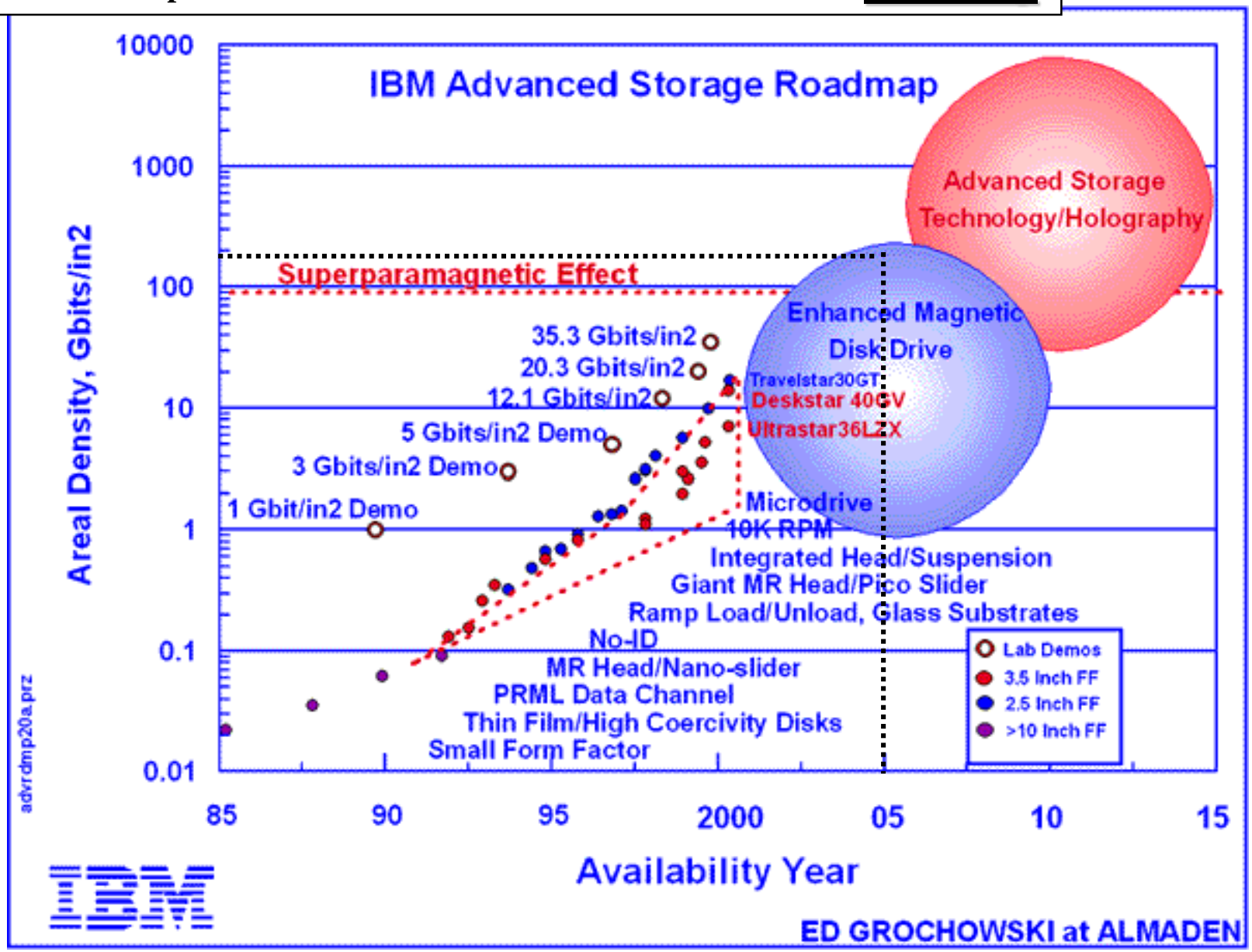



**Cost Per MByte:**
**> 100,000X times cost drop**

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 (Fourth Quarter 2014):**
**~ 0.00005 dollars per MByte or about 20 GBytes /Dollar**

**Historic Perspective of Hard Drive Characteristics Evolution: Roadmap**




**Current Areal Density ~ 640 Gbits / In$^2$**

# 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:**

| CPU | Memory | I/O |
|---|---|---|

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