

Lecture 2

Metrics of Storage Quality

Outline

- Storage system performance
 - Throughputs and latencies
 - Relationship to workload
- Overall system performance
 - Understanding the relationship
- Dependability
 - Reliability and availability
- Other important metrics
 - Cost and capacity
 - Manageability
 - Predictability
 - Power dissipation
 - Acoustics

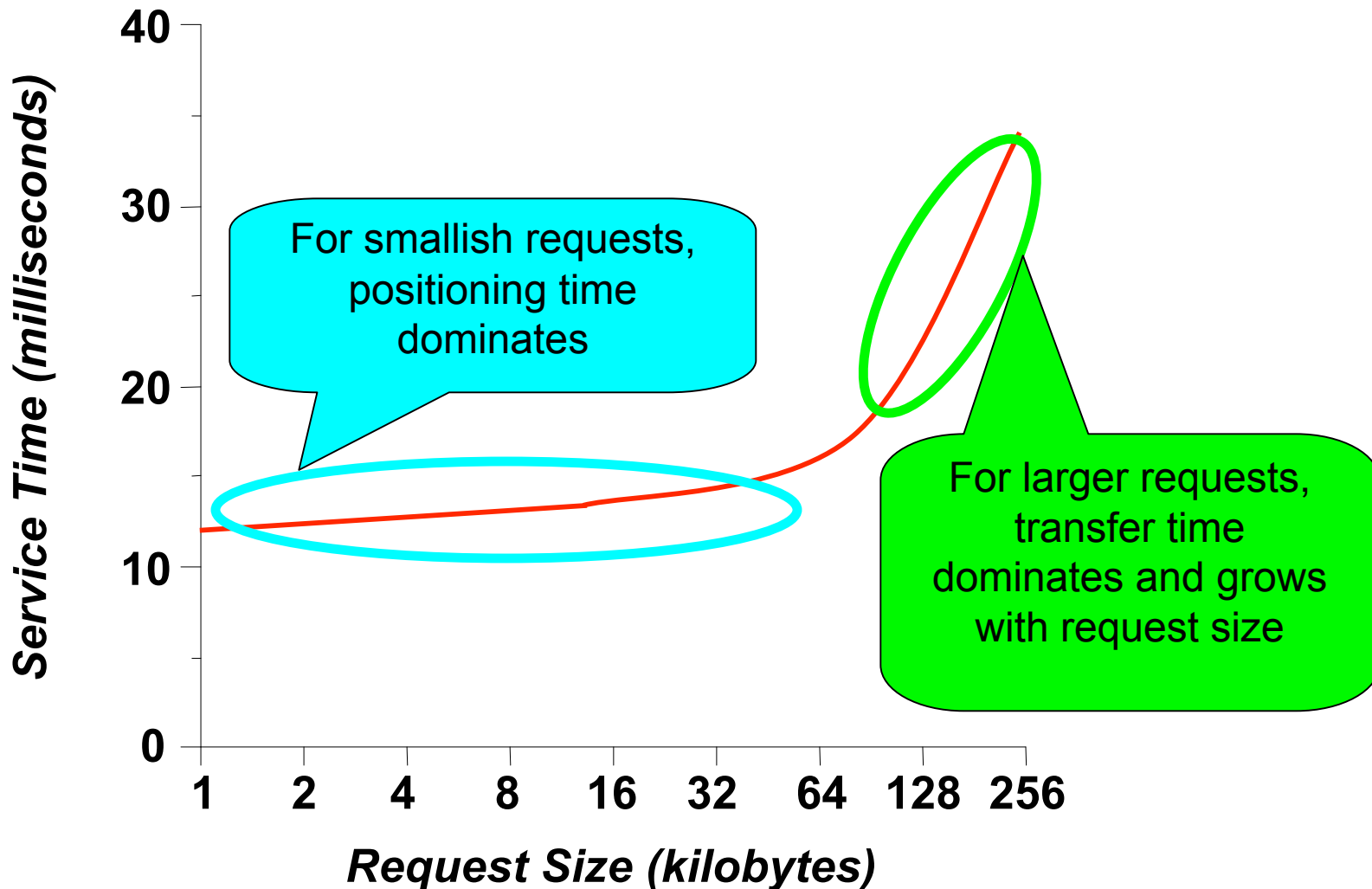
Basic Storage Metrics

- Bandwidth: rate of data movement
 - Generally specified as data units per unit of time
 - e.g., KB/s or Gbit/s
- Throughput: work units per unit time
 - e.g., accesses/second, IOs/s, requests/second, ...

Throughput versus bandwidth

- Observed bandwidth rarely matches peak bandwidth
- Why?
 - there is more to servicing a request than just data transfer
 - example: mechanical positioning
 - example: bus arbitration

Example: Service time vs. Request size

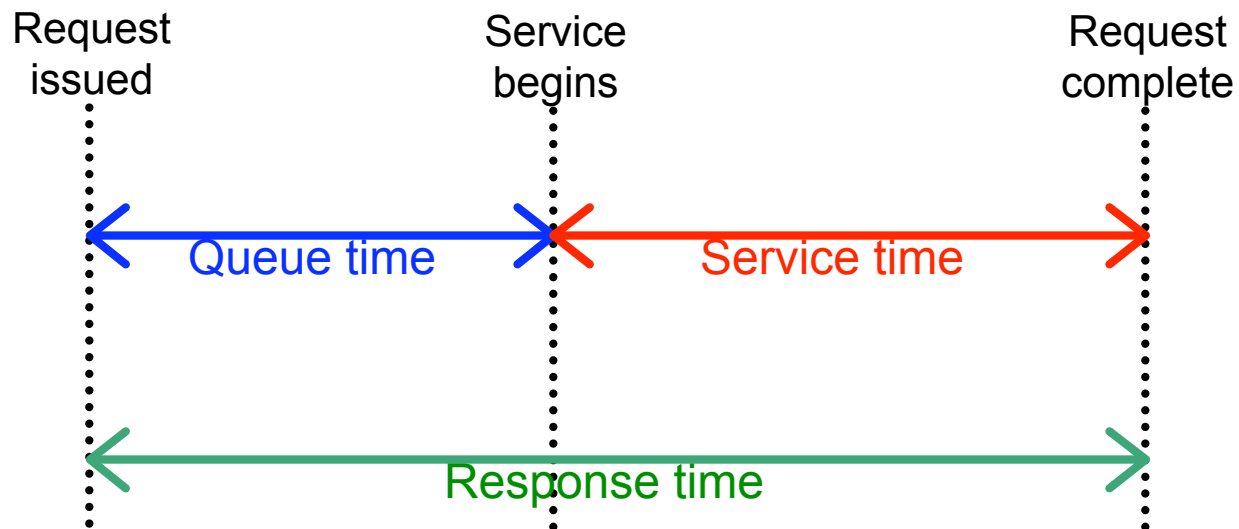


Bandwidth math (based on graph)

- Truly independent, random accesses have avg. seek
- For this disk
 - 4KB accesses take about 12.5 ms on average
 - so, 80 per second
 - $80/\text{second} * 4\text{KB} = 320 \text{ KB/second}$
 - 64KB accesses take about 15 ms on average
 - so, 67 per second
 - $67/\text{second} * 64\text{KB} = 4.3 \text{ MB/second}$
 - 256KB accesses take about 33 ms on average
 - so, 30 per second
 - $30/\text{second} * 7.7 \text{ MB/second} = 7.7 \text{ MB/second}$
 - Best case for this disk is about 8.5 MB/second

Basic Storage Latency Metrics

- Service time: time required to service I/O request
 - from start of service to completion
- Queue time: time spent waiting for turn to be serviced
- Response time = queue time + service time



Factors determining storage latencies

- Service time (T_S)
 - depends on system/device properties
 - e.g., acceleration speeds and data rates
 - depends on current status (and thus prior request stream)
 - e.g., cache state, disk head location
- Queue time (T_Q)
 - depends on service time (of course)
 - depends on request arrival pattern
 - inter-arrival time distribution
 - request rate = $1 / (\text{average inter-arrival time})$

Example inter-arrival time distribution: constant

- One new request every X ms ($T_{IA} = X$ ms)

- Assuming $T_{IA} \geq T_S$

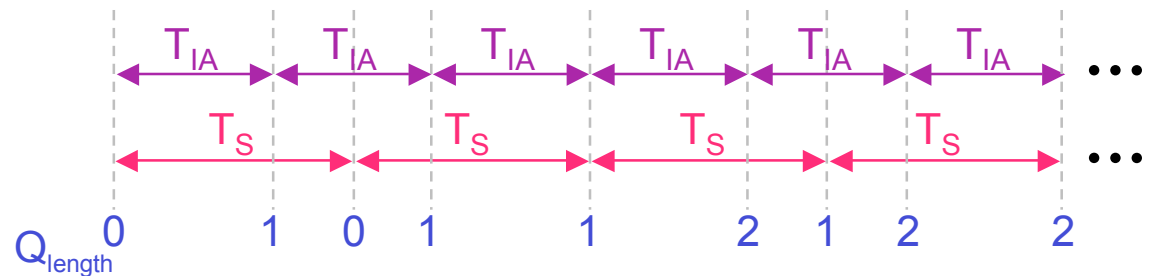
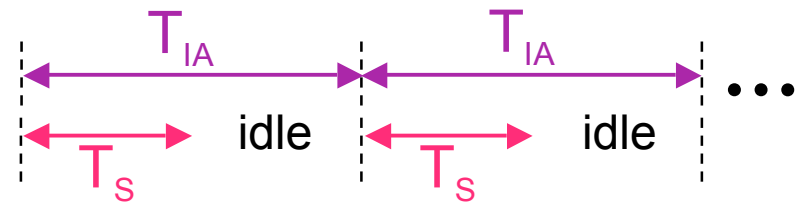
- Queue time = 0

- Assuming constant service time
- Variation could cause some queueing

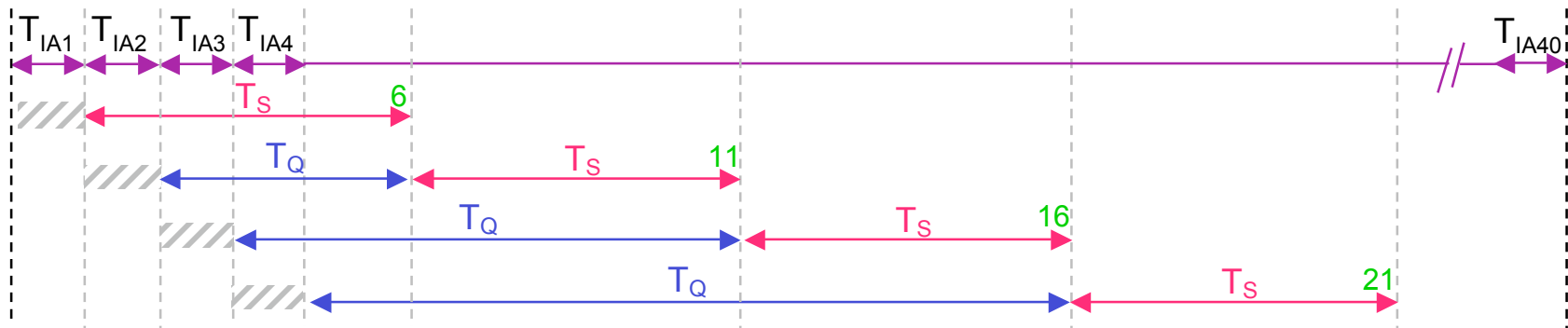
- Response time = T_S

- Assuming $T_{IA} < T_S$

- Explosion!



What is burstiness??



- Variation in inter-arrival times
 - yields more queuing delays for a given $|T_{IA}|$

Basic queuing theory

- Little's Law
 - # of requests outstanding = arrival rate * mean response time
 - sounds simple, but you'd be amazed at how long it took to prove
 - Very powerful tool for reasoning about performance
- For exponential RVs:

$$\text{Time}_{\text{queue}} = \text{Time}_{\text{server}} * \frac{\text{Server Utilization}}{(1 - \text{Server Utilization})}$$

- For 20ms service times and 10 I/Os per second, average response time is 25ms (assuming exponential distribution)

Doubling service capacity

- Doubling server speed to 10ms
 - $\text{Time}_{\text{queue}} = \text{Time}_{\text{server}} * (\text{util} / (1 - \text{util}))$
 - $= 10\text{ms} * (0.1 / 0.9) = 1.1 \text{ ms}$
 - $\text{Response time} = 10 \text{ ms} + 1.1 \text{ ms} = 11.1 \text{ ms}$
- Doubling number of servers (still at 20ms)
 - $\text{Time}_{\text{queue}} = \text{Time}_{\text{server}} * (\text{util} / (1 - \text{util}))$
 - $= 20\text{ms} * (0.1 / 0.9) = 2.2 \text{ ms}$
 - $\text{Response time} = 20 \text{ ms} + 2.2 \text{ ms} = 22.2 \text{ ms}$
- Non-linear scaling
 - bigger computers are different from more computers

Storage workloads

- Definition: “storage workload”
 - sequence of requests and their arrival times
 - note: not independent of storage performance
- What do storage workloads look like?

Conventional wisdom about workloads

- Nothing is constant, everything varies
 - request sizes, inter-arrival times, read:write, locations
- Access patterns exhibit locality of reference
 - both temporal and spatial
- Inter-arrival patterns are highly bursty
 - plenty of queueing and plenty of idle time
- All of this varies over time as well
 - phased behavior for a given workload
- And, it all depends on the particular workload!!

Key applications and some characteristics

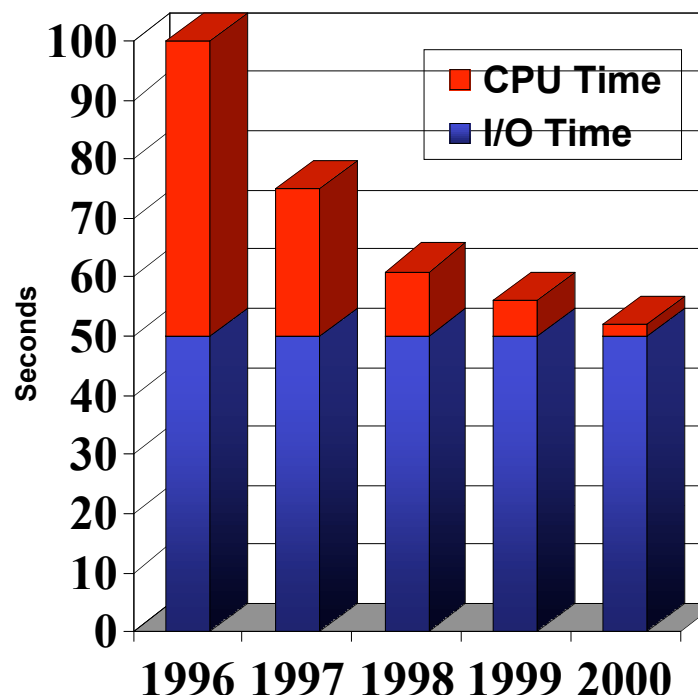
- Transaction processing (e.g., banks and e-commerce)
 - small requests (database page size: 2KB or 4KB)
 - read-modify-write pattern, but otherwise little locality
 - many I/Os per second
- Video playback (e.g., video on demand, DVR)
 - large-ish requests (video segment: 64KB-256KB)
 - predictability and bandwidth are key
- Scientific computing
 - very large, sequential reads and writes
 - bandwidth is key
- File servers
 - mixed, but mostly small reads/writes with locality

Context Switch!

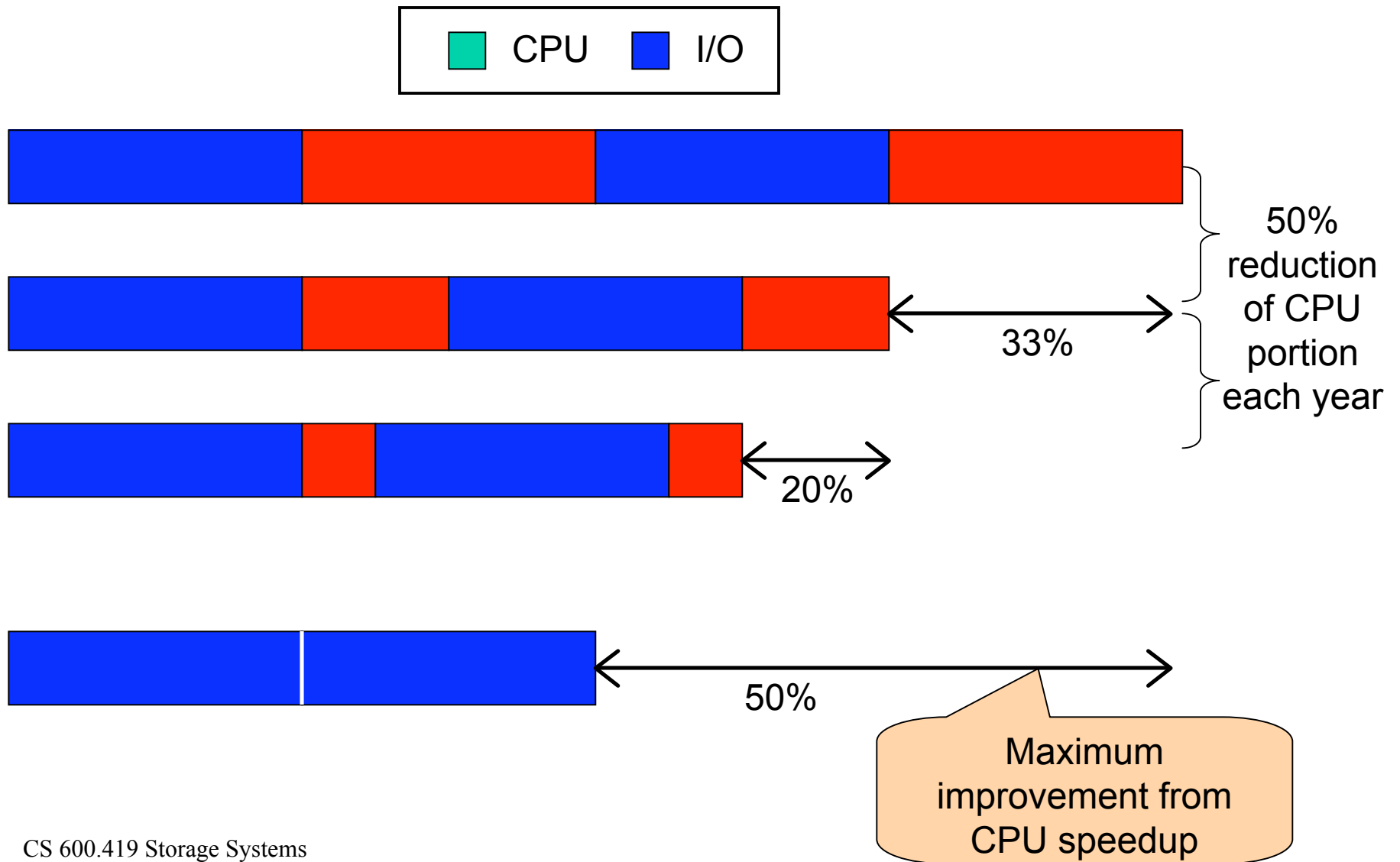
- Now, let's talk about how storage I/O performance relates to overall system performance

Remember this from last time?

- Assume 50 seconds CPU & 50 seconds I/O
- 1998 - 1999
 - CPU improves by: $N = 50/25 = 2$
 - Program performance improves by:
 $N = 100/75 = 1.33$
- 1999 - 2000
 - CPU performance - factor of 2
 - Program performance $N = 75/62.5 = 1.2$
- 2000 - 2001
 - CPU performance - factor of 2
 - Program performance $N = 62.5/56.5 = 1.11$
- Example of Amdahl's Law



One application pattern: read&process cycle



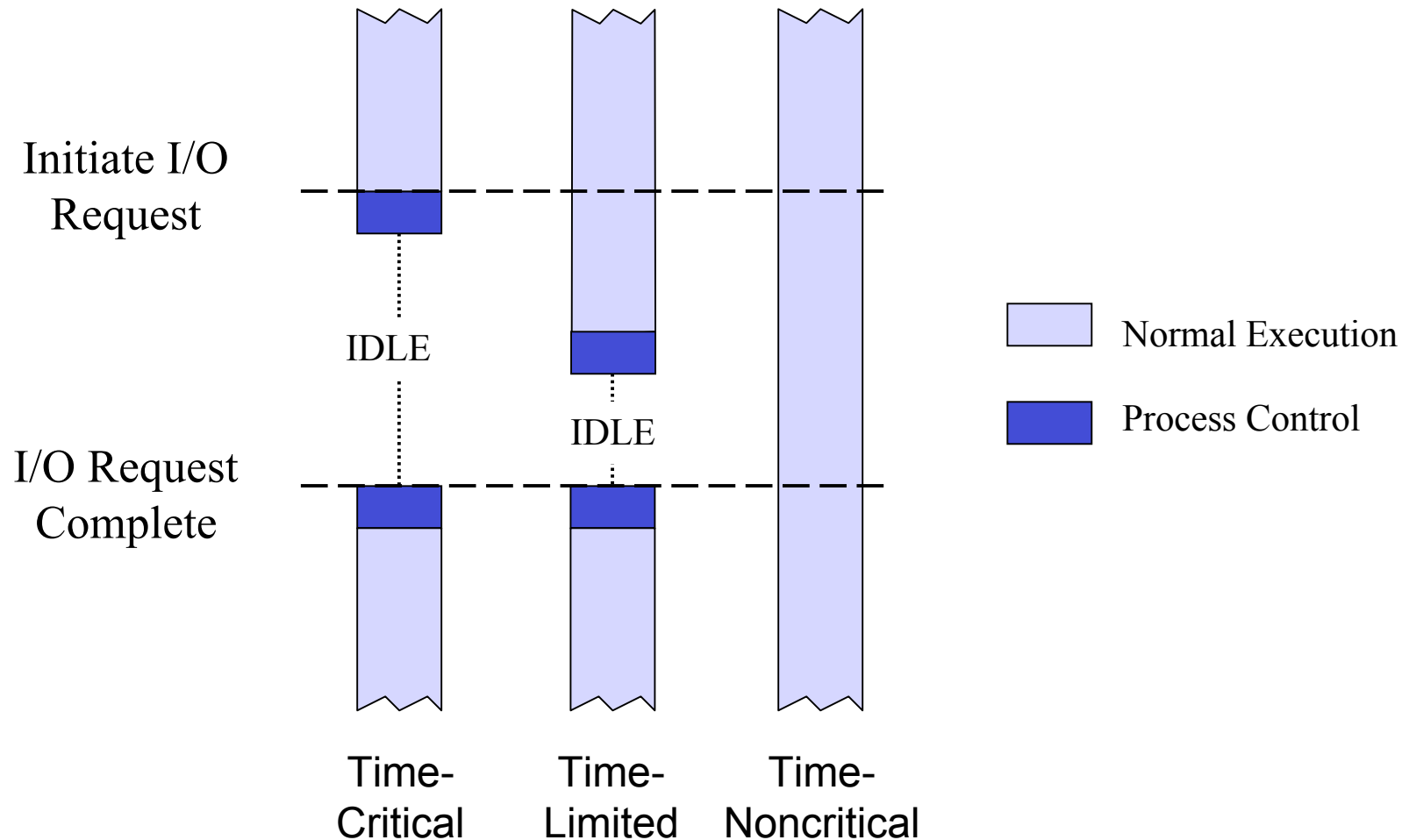
How about adding processes?

- I/O latencies are obviously a problem
- However, when one process stops to wait, another one can utilize the now-available CPU cycles
 - For this reason, some argue that I/O performance is not such a big issue
 - If there are enough active processes in the system, it's true
 - but, each year, the number of processes needed grows...
- Note: this only works so long as the problem is I/O latency rather than I/O throughput
 - Once I/O throughput becomes the problem, further processes generating requests will just have to wait longer for service...

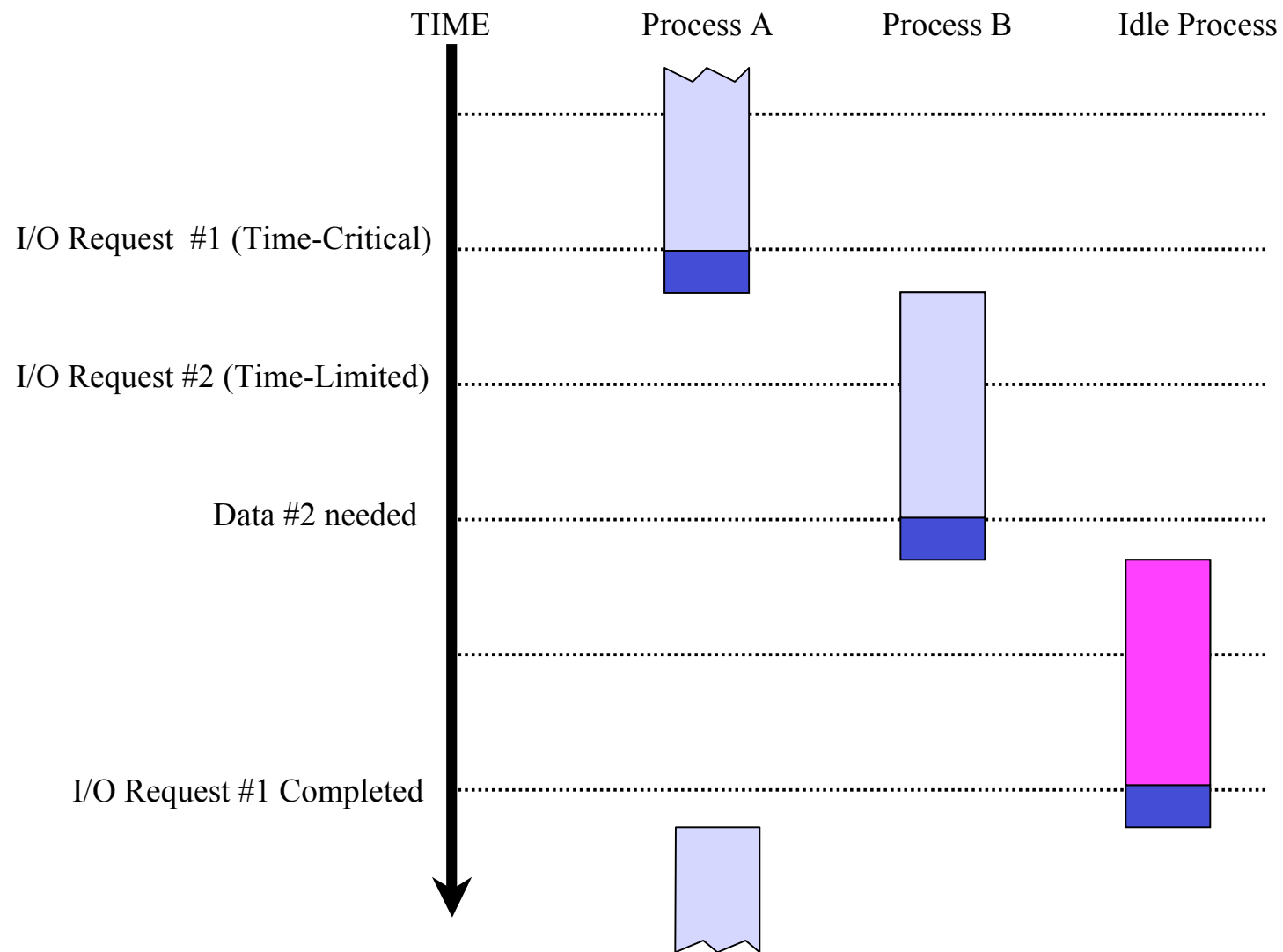
Request criticality

- Time-critical requests
 - issuing process blocks immediately and waits
 - example: page faults
- Time-limited requests
 - issuing process blocks if request takes too long
 - “too long” represented by associated *time limit*
 - example: prefetches
- Time-noncritical requests
 - issuing process will not block because of request
 - example: background cache write-backs

Illustration of request criticalities



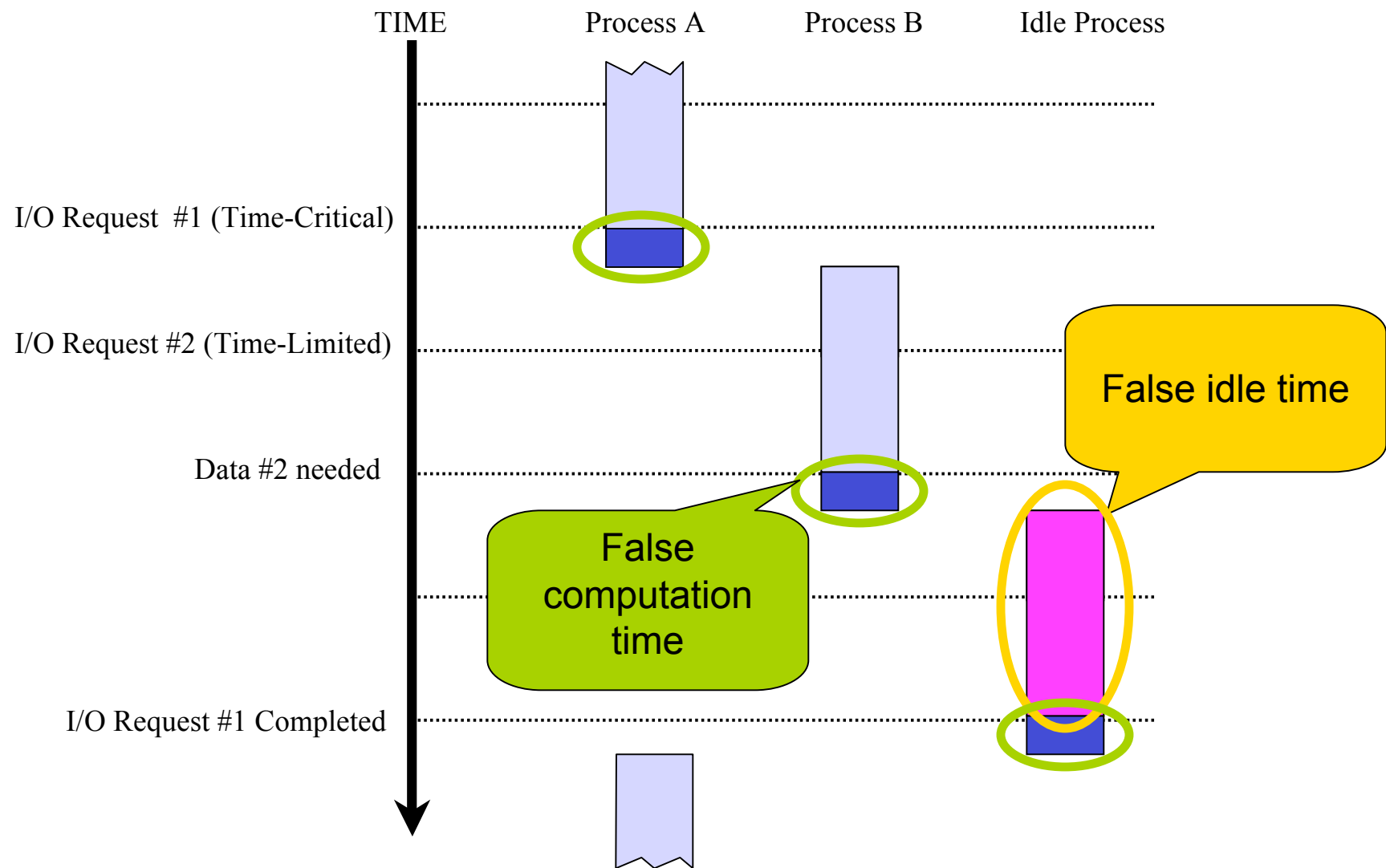
Process-level activity in a system



Some system-level impacts of I/O

- False Idle Time
 - Idle CPU time when all active processes wait for I/O
- False Computation Time
 - Wasted CPU time caused by avoiding false idle time
 - Process Control
 - Context Switches
- Resource Contention for I/O request processing
 - CPU Time
 - Interrupt processing
 - Managing storage controller interactions
 - System Bus and Memory Bus
 - Storage data transfers versus CPU loads/stores

Let's identify the I/O-induced delays

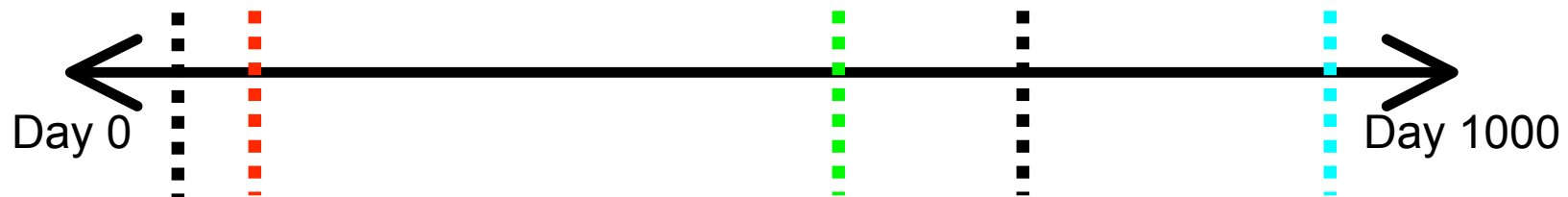


Now for some other metrics

- Dependability metrics
 - Reliability
 - Availability
- Cost metrics
 - Capacity
 - Manageability
- Other important metrics
 - Predictability
 - Power dissipation
 - Acoustics

Reliability

- Mean Time Between Failures (MTBF)
 - Usually computed by dividing a length of time by the number of failures during that time (averaged over a large population of items)



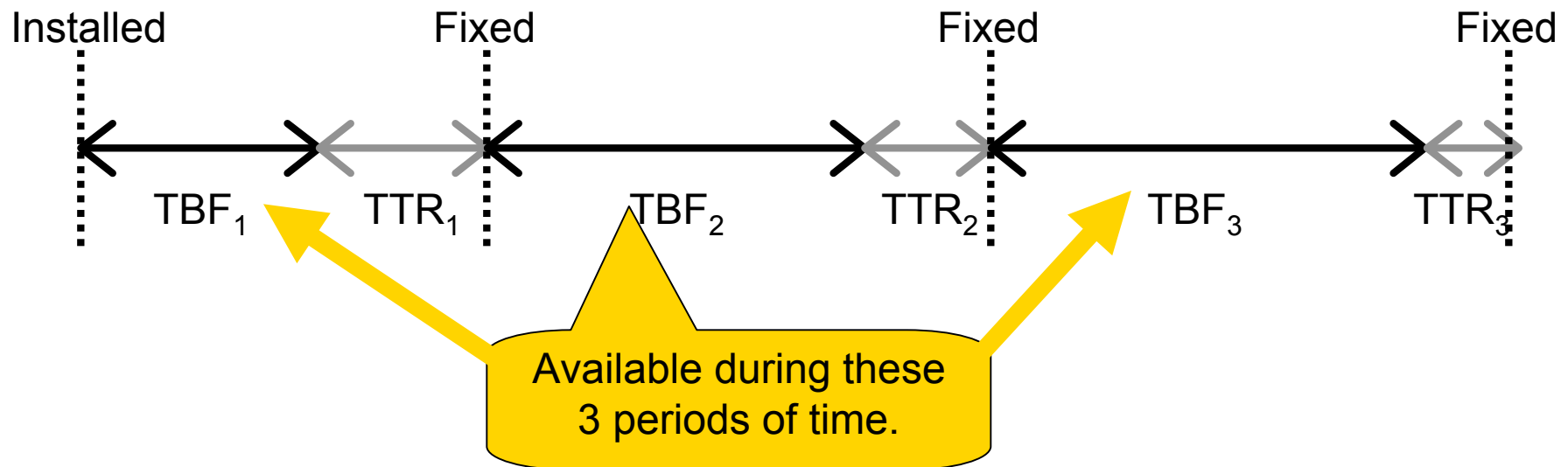
$$\text{MTBF} = \frac{1000 \text{ Days}}{5 \text{ failures} / 10 \text{ devices}} = 2000 \text{ Days [per device]}$$

- Note: NOT a guaranteed lifetime for a particular item!

Availability

- Fraction of time that server is able to handle requests
 - Computed from MTBF and MTTR (Mean Time To Repair)

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

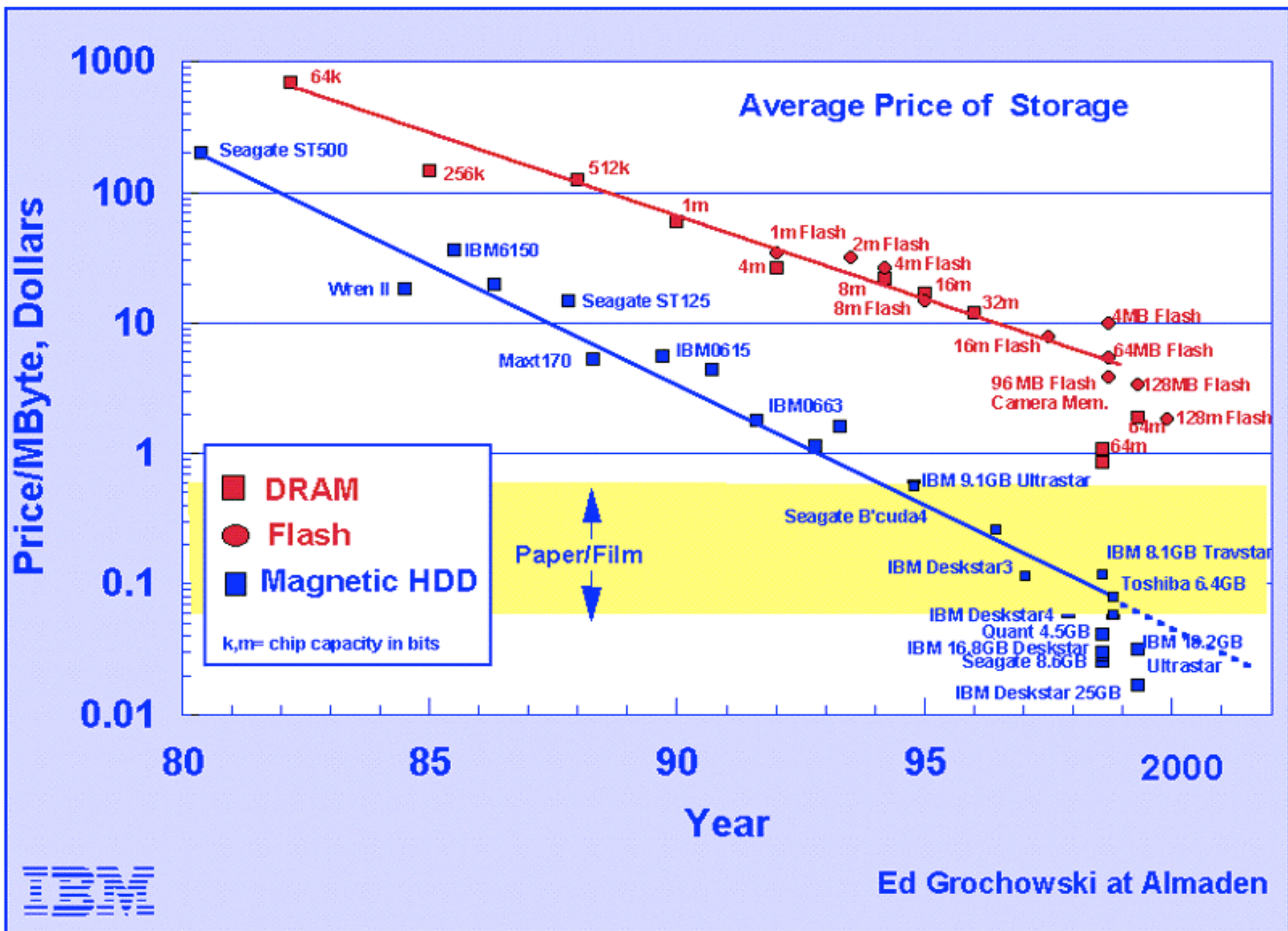


Availability versus Reliability

- What are the implications of:
 - a highly available system that is not highly reliable?
 - a highly reliable system that is not highly available?
- Researchers tend to study reliability
 - arithmetic rather than differential equations
 - but, it is frequently not the right metric

Capacity Metrics

- Capacity = how much stuff can be held
- Good news -- exponential growth in hardware capacity
 - Logic growth: 60% to 80% per year
 - DRAM growth: 60% per year (in 400% increments/3 years)
 - Disk growth: 100% per year (was 60%/year in mid-90s)
- Bad news -- exponential growth in hardware usage
 - e.g., program size: 50% to 100% per year
 - data mining records
 - no more pruning of personal files
- Consequence
 - Storage may be approaching “free” on a per-bit basis, but somehow it always seems to fill up



BUT, cost is more than just storage capacity

- Need to connect storage devices to a system
 - Interconnect cables
 - Power, heat dissipation, etc.
 - Storage controller cards
- Need to manage the storage
 - People
 - Software
- For client/server, need server system
 - ... and a single server can only support so much storage

Manageability

- Very difficult to quantify
 - perhaps hours of sysadmin time required?
- Some of the major management tasks
 - Back-up, restore, and disaster protection
 - Capacity planning and load balancing
 - Failure detection and repair
 - Quota management
 - Installation and upgrades

New metrics of importance

- Predictability
 - particularly important to emerging video and audio apps
 - e.g., mp3 players and DVR boxes
 - is sometimes quantified by response time variance
 - But, response times can be predictable even when they vary

New metrics of importance

- Predictability
 - particularly important to emerging video and audio apps
 - e.g., mp3 players and TiVo boxes
 - is sometimes quantified by response time variance
 - But, response times can be predictable even when they vary
- Power dissipation
 - particularly important to mobile computing environments
 - e.g., laptop computers and PDAs
 - usually quantified by Watts
 - correct metric for heat-related issues
 - for battery lifetime, Joules ($= \text{Watts} * \text{time}$) is what matters

New metrics of importance

- Predictability
 - particularly important to emerging video and audio apps
 - e.g., mp3 players and TiVo boxes
 - is sometimes quantified by response time variance
 - But, response times can be predictable even when they vary
- Power dissipation
 - particularly important to mobile computing environments
 - e.g., laptop computers and PDAs
 - usually quantified by Watts
 - correct metric for heat-related issues
 - for battery lifetime, Joules ($= \text{Watts} * \text{time}$) is what matters
- Acoustics
 - Important for home and office environments
 - Usually quantified by decibels (peak, average, and changes)
 - Average is least problematic, since it can be white noise