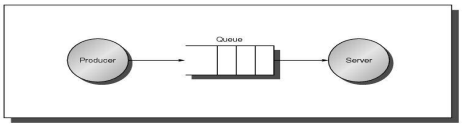
Storage Systems (2)

Outline

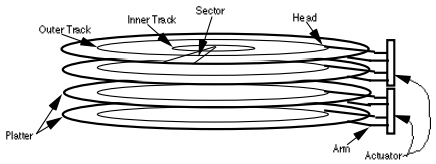
- Historical Context of Storage I/O
- Secondary and Tertiary Storage Devices
- I/O Buses
- Processor Interface Issues
- **Storage I/O Performance Measures**
- ***A Little Queuing Theory***
- ***Redundant Arrarys of Inexpensive Disks (RAID)***

I/O Performance Measures

- **Diversity**
  - **Which IO devices can connect?**
- **Capacity**
  - **How many IO devices can connect?**
- **Response time (Latency)**
  - **time a task takes from the moment it is placed in the queue until the service is completed by the server**
- **Throughput (IO Bandwidth)**
  - **average number of tasks completed by the server over a time period**



Disk Device Performance



Disk Latency = Queuing Time + Seek Time + Rotation Time + Xfer Time

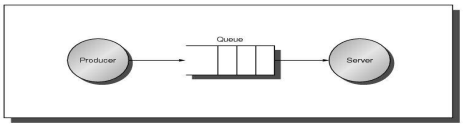
Order of magnitude times for 4K byte transfers:

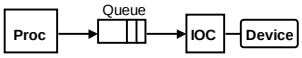
Seek: 12 ms or less

Rotate: 4.2 ms @ 7200 rpm (8.3 ms @ 3600 rpm)

Xfer: 1 ms @ 7200 rpm (2 ms @ 3600 rpm)

Disk I/O Performance





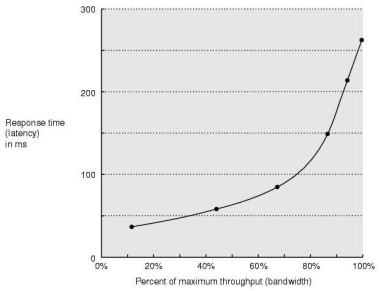
Response time = Queue + Device Service time

Disk Latency = Queuing Time + Controller Overhead + Seek Time + Rotation Time + Transfer Time

Disk Time Example

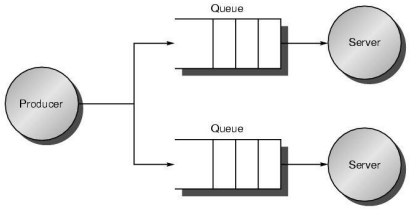
- **Disk Parameters:**
  - **Transfer size is 8K bytes**
  - **Advertised average seek is 12 ms**
  - **Disk spins at 7200 RPM**
  - **Transfer rate is 4 MB/sec**
- **Controller overhead is 2 ms**
- **Assume that disk is idle so no queuing delay**
- **What is Average Disk Access Time for a Sector?**
  - **Ave seek + ave rot delay + transfer time + controller overhead**
  - **12 ms + 0.5/(7200 RPM/60) + 8 KB/4 MB/s + 2 ms**
  - **12 + 4.15 + 2 + 2 = 20 ms**
- **Advertised seek time assumes no locality: typically 1/4 to 1/3 advertised seek time: 20 ms ➔ 12 ms**

Response Time vs Throughput



Improving Performance I

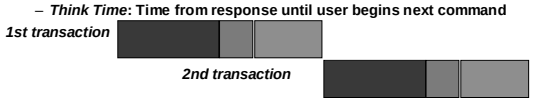
- **Provide more resources**
  - **Improve throughput and response time?**



Improving Performance II Response Time vs. Productivity

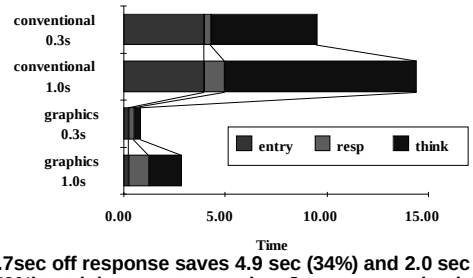
- **Interactive environments:**

  Each interaction or *transaction* has 3 parts:
  - ***Entry Time*: time for user to enter command**
  - ***System Response Time*: time between user entry & system reply**
  - ***Think Time*: Time from response until user begins next command**



- **What happens to transaction time as shrink system response time from 1.0 sec to 0.3 sec?**
  - **With Keyboard: 4.0 sec entry, 9.4 sec think time**
  - **With Graphics: 0.25 sec entry, 1.6 sec think time**

Response Time & Productivity



- **0.7sec off response saves 4.9 sec (34%) and 2.0 sec (70%) total time per transaction → greater productivity**
- **Another study: everyone gets more done with faster response, but novice with fast response = expert with slow**

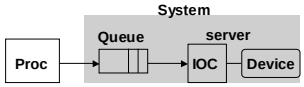
Outline

- Historical Context of Storage I/O
- Secondary and Tertiary Storage Devices
- I/O Buses
- Processor Interface Issues
- ***Storage I/O Performance Measures***
- **A Little Queuing Theory**
- ***Redundant Arrays of Inexpensive Disks (RAID)***

Introduction to Queueing Theory

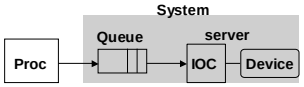
- **More interested in long term, steady state than in startup → Arrivals = Departures**
- **Little's Law:**
  Mean number tasks in system
  = arrival rate x mean response time
  - **Observed by many, Little was first to prove**
- **Applies to any system in equilibrium, as long as nothing in black box is creating or destroying tasks**

A Little Queuing Theory: Notation



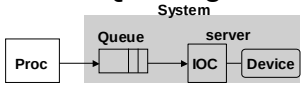
- **Queuing models assume state of equilibrium: input rate = output rate**
- **Notation:**
  - r (λ) average number of arriving customers/second
  - T_{ser} average time to service a customer (traditionally $\mu = 1/T_{ser}$)
  - u (ρ) server utilization (0..1): $u = r \times T_{ser}$ (or $u = r / \mu$)
  - T_q average time/customer in queue
  - T_{sys} average time/customer in system: $T_{sys} = T_q + T_{ser}$
  - L_q average length of queue: $L_q = r \times T_q$
  - L_{sys} average length of system: $L_{sys} = r \times T_{sys}$
- **Little's Law: $Length_{system} = rate \times Time_{system}$**
  (Mean number customers = arrival rate x mean service time)

A Little Queuing Theory



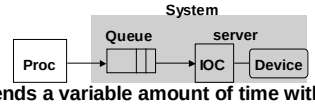
- **Service time completions vs. waiting time for a busy server: randomly arriving event joins a queue of arbitrary length when server is busy, otherwise serviced immediately**
  - **Unlimited length queues key simplification**
- **A *single server queue*: combination of a servicing facility that accommodates 1 customer at a time (*server*) + waiting area (*queue*): together called a *system***
- **Server spends a variable amount of time with customers; *how do you characterize variability?***
  - **Distribution of a random variable: histogram? curve?**
  - **Mean and variance sufficient to characterize distribution**

A Little Queuing Theory



- **Server spends a variable amount of time with customers**
  - **Weighted mean *m1* = (f1 x T1 + f2 x T2 +...+ fn x Tn)/F (F=f1 + f2...)**
  - ***variance* = (f1 x $T1^2$ + f2 x $T2^2$ +...+ fn x Tn^2)/F – $m1^2$**
    - » **Must keep track of unit of measure (100 ms^2 vs. 0.1 s^2)**
  - ***Squared coefficient of variance*: C = variance/$m1^2$**
    - » **Unitless measure (100 ms^2 vs. 0.1 s^2)**
- **Exponential distribution *C = 1* : most short relative to average, few others long; 90% < 2.3 x average, 63% < average**
- **Hypoexponential distribution *C < 1* : most close to average, C=0.5 => 90% < 2.0 x average, only 57% < average**
- **Hyperexponential distribution *C > 1* : further from average C=2.0 => 90% < 2.8 x average, 69% < average**

A Little Queuing Theory: Variable Service Time



- **Server spends a variable amount of time with customers**
  - **Weighted mean *m1* = (f1xT1 + f2xT2 +...+ fnxTn)/F (F=f1+f2+...)**
  - **Squared coefficient of variance *C***
- **Disk response times *C = 1.5* (majority seeks < average)**
- **Yet usually pick *C = 1.0* for simplicity**
- **Another useful value is average time must wait for server to complete task: *m1(z)***
  - **Not just 1/2 x m1 because doesn't capture variance**
  - **Can derive *m1(z) = 1/2 x m1 x (1 + C)***
  - ***No variance → C= 0 → m1(z) = 1/2 x m1***

A Little Queuing Theory: Average Wait Time

- Calculating average wait time in queue T_q
  - If something at server, it takes to complete on average *m1(z)*
  - Chance server is busy = u; average delay is *u* x *m1(z)*
  - All customers in line must complete; each avg T_{ser}

  $T_q = u \times m1(z) + L_q \times T_{ser} = 1/2 \times u \times T_{ser} \times (1 + C) + L_q \times T_{ser}$
  $T_q = 1/2 \times u \times T_{ser} \times (1 + C) + r \times T_q \times T_{ser}$
  $T_q = 1/2 \times u \times T_{ser} \times (1 + C) + u \times T_q$
  $T_q \times (1 - u) = T_{ser} \times u \times (1 + C) /2$
  $T_q = T_{ser} \times u \times (1 + C) / (2 \times (1 - u))$
- Notation:

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

A Little Queuing Theory: M/G/1 and M/M/1

- Assumptions so far:
  - System in equilibrium
  - Time between two successive arrivals in line are random
  - Server can start on next customer immediately after prior finishes
  - No limit to the queue: works First-In-First-Out
  - Afterward, all customers in line must complete; each avg T_{ser}
- Described "memoryless" or Markovian request arrival (M for C=1 exponentially random), General service distribution (no restrictions), 1 server: *M/G/1 queue*
- When Service times have C = 1, *M/M/1 queue*
  $T_q = T_{ser} \times u \times (1 + C) /(2 \times (1 - u)) = T_{ser} \times u / (1 - u)$

T_{ser}	average time to service a customer
u	server utilization (0..1): $u = r \times T_{ser}$
T_q	average time/customer in queue

A Little Queuing Theory: An Example

- processor sends 10 x 8KB disk I/Os per second, requests & service exponentially distrib., avg. disk service = 20 ms
- On average, how utilized is the disk?
  - What is the number of requests in the queue?
  - What is the average time spent in the queue?
  - What is the average response time for a disk request?
- Notation:

r	average number of arriving customers/second = 10
T_{ser}	average time to service a customer = 20 ms (0.02s)
u	server utilization (0..1): $u = r \times T_{ser}$ = 10/s x .02s = 0.2
T_q	average time/customer in queue = $T_{ser} \times u / (1 - u)$ = 20 x 0.2/(1-0.2) = 20 x 0.25 = 5 ms (0 .005s)
T_{sys}	average time/customer in system: $T_{sys} = T_q + T_{ser}$ = 25 ms
L_q	average length of queue: $L_q = r \times T_q$ = 10/s x .005s = 0.05 requests in queue
L_{sys}	average # tasks in system: $L_{sys} = r \times T_{sys}$ = 10/s x .025s = 0.25

A Little Queuing Theory: Another Example

- processor sends 20 x 8KB disk I/Os per sec, requests & service exponentially distrib., avg. disk service = 12 ms
- On average, how utilized is the disk?
  - What is the number of requests in the queue?
  - What is the average time a spent in the queue?
  - What is the average response time for a disk request?
- Notation:

r	average number of arriving customers/second= 20
T_{ser}	average time to service a customer= 12 ms
u	server utilization (0..1): $u = r \times T_{ser}$ = ___/s x .___s =
T_q	average time/customer in queue = $T_{ser} \times u / (1 - u)$ = ___x ___/(_____) = ___ x ____ = ___ ms
T_{sys}	average time/customer in system: $T_{sys} = T_q + T_{ser}$ = 16 ms
L_q	average length of queue: $L_q = r \times T_q$ = ___/s x ____s = ____ requests in queue
L_{sys}	average # tasks in system : $L_{sys} = r \times T_{sys}$ = ____/s x__ s =

A Little Queuing Theory: Another Example

- processor sends 20 x 8KB disk I/Os per sec, requests & service exponentially distrib., avg. disk service = 12 ms
- On average, how utilized is the disk?
  - What is the number of requests in the queue?
  - What is the average time a spent in the queue?
  - What is the average response time for a disk request?
- Notation:

r	average number of arriving customers/second= 20
T_{ser}	average time to service a customer= 12 ms
u	server utilization (0..1): $u = r \times T_{ser}$= 20/s x .012s = 0.24
T_q	average time/customer in queue = $T_{ser} \times u / (1 - u)$ = 12 x 0.24/(1-0.24) = 12 x 0.32 = 3.8 ms
T_{sys}	average time/customer in system: $T_{sys} = T_q + T_{ser}$= 15.8 ms
L_q	average length of queue:$L_q = r \times T_q$ = 20/s x .0038s = 0.076 requests in queue
L_{sys}	average # tasks in system : $L_{sys} = r \times T_{sys}$ = 20/s x .016s = 0.32

A Little Queuing Theory: Yet Another Example

- Suppose processor sends 10 x 8KB disk I/Os per second, squared coef. var.(C) = 1.5, avg. disk service time = 20 ms
- On average, how utilized is the disk?
  - What is the number of requests in the queue?
  - What is the average time a spent in the queue?
  - What is the average response time for a disk request?
- Notation:

r	average number of arriving customers/second= 10
T_{ser}	average time to service a customer= 20 ms
u	server utilization (0..1): $u = r \times T_{ser}$= 10/s x .02s = 0.2
T_q	average time/customer in queue = $T_{ser} \times u \times (1 + C) / (2 \times (1 - u))$ = 20 x 0.2(2.5)/2(1 – 0.2) = 20 x 0.32 = 6.25 ms
T_{sys}	average time/customer in system: $T_{sys} = T_q + T_{ser}$= 26 ms
L_q	average length of queue:$L_q = r \times T_q$ = 10/s x .006s = 0.06 requests in queue
L_{sys}	average # tasks in system :$L_{sys} = r \times T_{sys}$ = 10/s x .026s = 0.26

Outline

- Historical Context of Storage I/O
- Secondary and Tertiary Storage Devices
- I/O Buses
- Processor Interface Issues
- ***Storage I/O Performance Measures***
- ***A Little Queuing Theory***
- **Redundant Arrays of Inexpensive Disks (RAID)**

Reliability, Availability, and RAID

- **Reliability**
  - Is anything broken?
- **Availability**
  - Is the system still available to the user?
- **Data integrity**
  - Is data consistent even in the event of failure, and hence data loss?

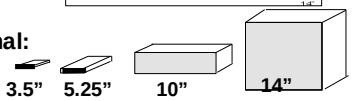
Improving availability and performance

➔ **disk arrays (striping of data over multiple disks)**

Manufacturing Advantages of Disk Arrays

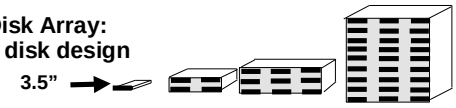
Disk Product Families

Conventional: 4 disk designs



Low End → High End

Disk Array: 1 disk design



Replace Small # of Large Disks with Large # of Small Disks! (1988 Disks)

	IBM 3390 (K)	IBM 3.5" 0061	*x70*
Data Capacity	20 GBytes	320 MBytes	*23 GBytes*
Volume	97 cu. ft.	0.1 cu. ft.	*11 cu. ft.*
Power	3 KW	11 W	*1 KW*
Data Rate	15 MB/s	1.5 MB/s	*120 MB/s*
I/O Rate	600 I/Os/s	55 I/Os/s	*3900 IOs/s*
MTTF	250 KHrs	50 KHrs	*??? Hrs*
Cost	$250K	$2K	*$150K*

Disk Arrays have potential for
- large data and I/O rates
- high MB per cu. ft., high MB per KW
- reliability?

Array Reliability

- Reliability of N disks = Reliability of 1 Disk ÷ N
  - 50,000 Hours ÷ 70 disks = 700 hours
  - Disk system MTTF: Drops from 6 years to 1 month!
- Arrays (without redundancy) too unreliable to be useful!

Hot spares support reconstruction in parallel with access: very high media availability can be achieved

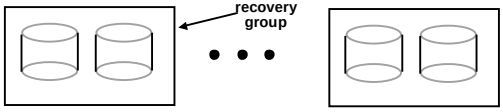
Redundant Arrays of Disks

- Files are "striped" across multiple spindles
- Redundancy yields high data availability
  - Disks will fail
  - Contents reconstructed from data redundantly stored in the array
    - → Capacity penalty to store it
    - → Bandwidth penalty to update

Techniques:
- Mirroring/Shadowing (high capacity cost)
- Horizontal Hamming Codes (overkill)
- Parity & Reed-Solomon Codes
- Failure Prediction (no capacity overhead!)

VaxSimPlus — Technique is controversial

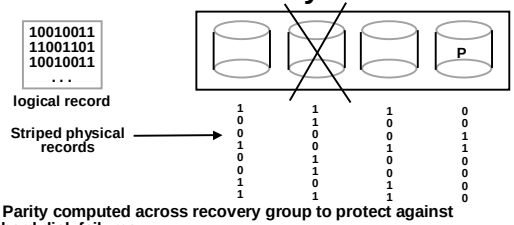
Redundant Arrays of Disks RAID 1: Disk Mirroring/Shadowing



- Each disk is fully duplicated onto its "shadow"
  Very high availability can be achieved
- Bandwidth sacrifice on write:
  Logical write = two physical writes
- Reads may be optimized
- Most expensive solution: 100% capacity overhead

Targeted for high I/O rate , high availability environments

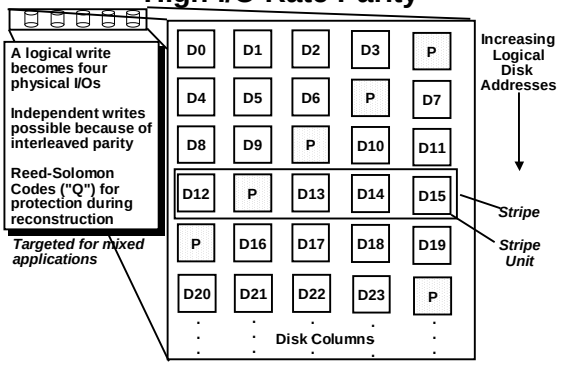
Redundant Arrays of Disks RAID 3: Parity Disk



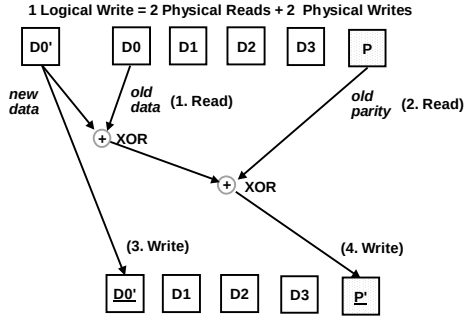
- Parity computed across recovery group to protect against hard disk failures
  33% capacity cost for parity in this configuration
  wider arrays reduce capacity costs, decrease expected availability, increase reconstruction time
- Arms logically synchronized, spindles rotationally synchronized
  logically a single high capacity, high transfer rate disk

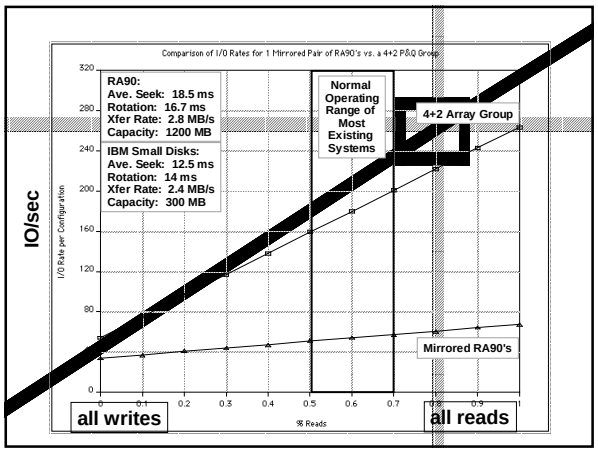
Targeted for high bandwidth applications: Scientific, Image Processing

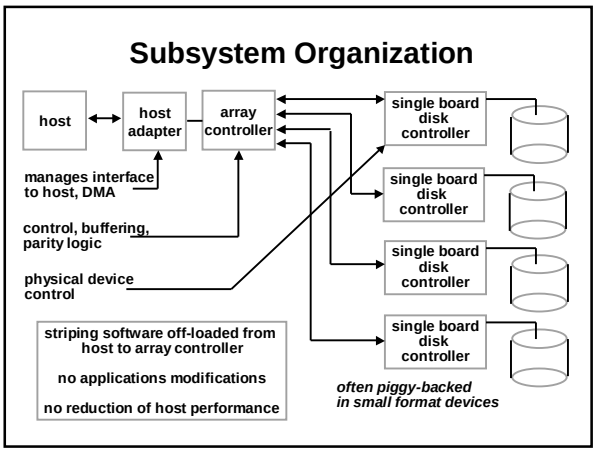
Redundant Arrays of Disks RAID 5+: High I/O Rate Parity

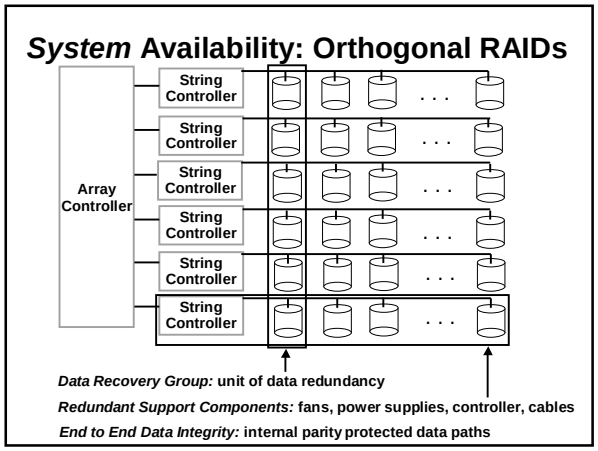


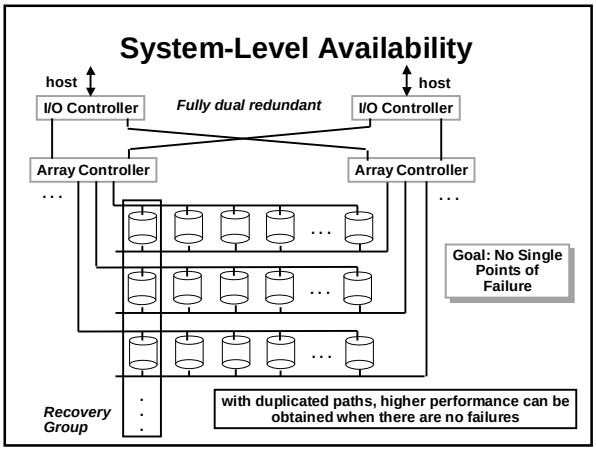
Problems of Disk Arrays: Small Writes




Comparison of I/O Rates for 1 Mirrored Pair of RA90's vs. a 4+2 P&Q Group
RA90:
Ave. Seek: 18.5 ms
Rotation: 16.7 ms
Xfer Rate: 2.8 MB/s
Capacity: 1200 MB
IBM Small Disks:
Ave. Seek: 12.5 ms
Rotation: 14 ms
Xfer Rate: 2.4 MB/s
Capacity: 300 MB
Normal Operating Range of Most Existing Systems
4+2 Array Group
Mirrored RA90's
IO/sec
I/O Rate per Configuration
% Reads
all writes
all reads

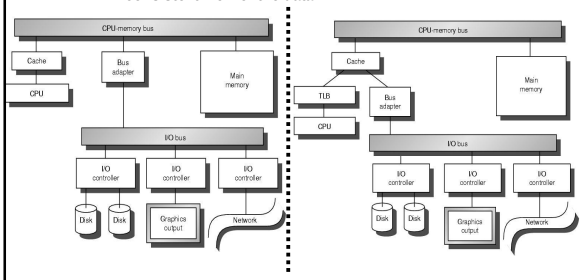

Subsystem Organization
host
host adapter
array controller
single board disk controller
single board disk controller
single board disk controller
single board disk controller
manages interface to host, DMA
control, buffering, parity logic
physical device control
striping software off-loaded from host to array controller
no applications modifications
no reduction of host performance
often piggy-backed in small format devices


System Availability: Orthogonal RAIDs
Array Controller
String Controller
String Controller
String Controller
String Controller
String Controller
String Controller
Data Recovery Group: unit of data redundancy
Redundant Support Components: fans, power supplies, controller, cables
End to End Data Integrity: internal parity protected data paths


System-Level Availability
host
host
I/O Controller
Fully dual redundant
I/O Controller
Array Controller
Array Controller
Goal: No Single Points of Failure
Recovery Group
with duplicated paths, higher performance can be obtained when there are no failures

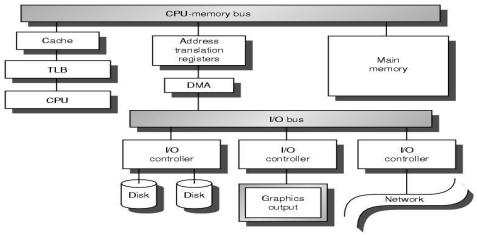
Interfacing to an Operating System

- Stale data problem
  - Inconsistent view of the data



DMA and Virtual Memory

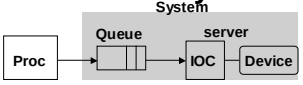
- Should DMA use virtual or physical address?



Chapter 6

- Did not discuss "Examples of Benchmarks of Disk Performance" on page 516
  - May be helpful in understanding context
  - Recommended reading, though will not be in exam
- Did not discuss Section 6.8
  - May be useful when studying Operating Systems
- Homework #5 due Dec. 10
  - Six questions in Section 6.7; note that answers are provided in text, but you still have to do it and SUBMIT them
  - 6.6, 6.10, 6.16, 6.17
  - Optional: 6.5, 6.7, 6.9

Summary: A Little Queuing Theory



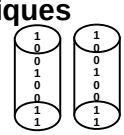
- Queuing models assume state of equilibrium: input rate = output rate
- Notation:

r	average number of arriving customers/second
T_{ser}	average time to service a customer (tradtionally $\mu = 1/T_{ser}$)
u	server utilization (0..1): $u = r \times T_{ser}$
T_q	average time/customer in queue
T_{sys}	average time/customer in system: $T_{sys} = T_q + T_{ser}$
L_q	average length of queue: $L_q = r \times T_q$
L_{sys}	average length of system : $L_{sys} = r \times T_{sys}$

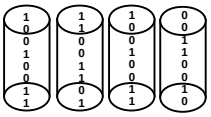
- Little's Law: $Length_{system} = rate \times Time_{system}$
  (Mean number customers = arrival rate x mean service time)

Summary: Redundant Arrays of Disks (RAID) Techniques

- *Disk Mirroring, Shadowing (RAID 1)*
  - Each disk is fully duplicated onto its "shadow"
  - Logical write = two physical writes
  - 100% capacity overhead



- *Parity Data Bandwidth Array (RAID 3)*
  - Parity computed horizontally
  - Logically a single high data bw disk



- *High I/O Rate Parity Array (RAID 5)*
  - Interleaved parity blocks
  - Independent reads and writes
  - Logical write = 2 reads + 2 writes
  - Parity + Reed-Solomon codes