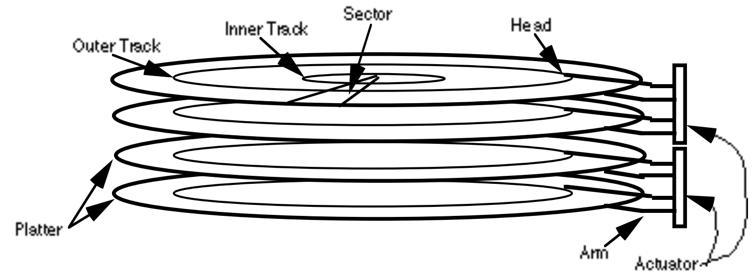
Lecture 13:
I/O: A Little Queuing Theory, RAID

Professor David A. Patterson
Computer Science 252
Spring 1998

Review: Disk Device Terminology



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)

Review

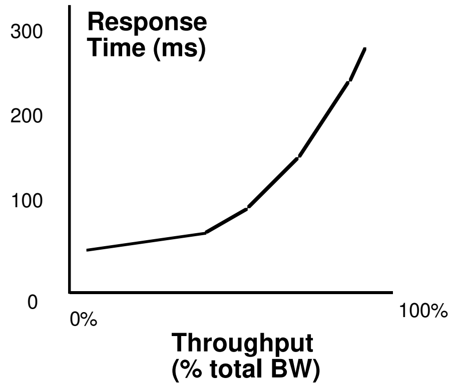
- Disk industry growing rapidly, improves:
  - bandwidth 40%/yr ,
  - areal density 60%/year, $/MB faster?
- queue + controller + seek + rotate + transfer
- Advertised average seek time benchmark much greater than average seek time in practice
- Response time vs. Bandwidth tradeoffs
- Value of faster response time:
  - 0.7sec off response saves 4.9 sec and 2.0 sec (70%) total time per transaction => greater productivity
  - everyone gets more done with faster response, but novice with fast response = expert with slow
- Processor Interface: today peripheral processors, DMA, I/O bus, interrupts

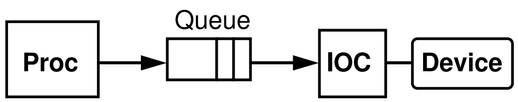
Review: Storage System Issues

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

Review: Disk I/O Performance

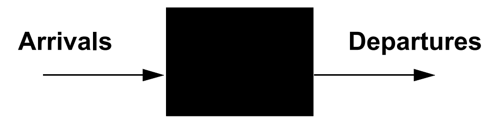
Metrics:
Response Time
Throughput





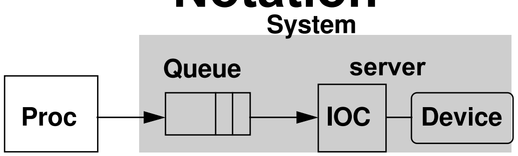
Response time = Queue + Device Service time

Introduction to Queueing Theory



- More interested in long term, steady state than in startup => Arrivals = Departures
- <u>Little's Law</u>:
  Mean number tasks in system = arrival rate x mean reponse 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

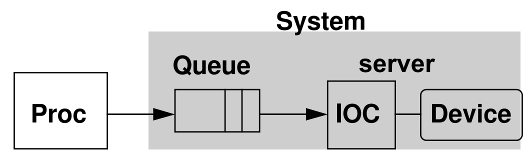
A Little Queuing Theory: Notation



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

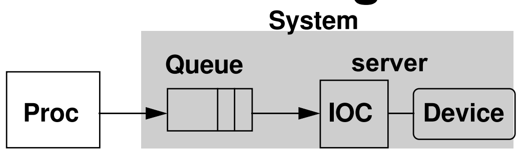
  r average number of arriving customers/second
  T_{ser} average time to service a customer (tradtionally $\mu = 1/ T_{ser}$)
  u server utilization (0..1): $u = r \times T_{ser}$ (or $u = r / T_{ser}$)
  T_q average time/customer in queue
  T_{sys} average time/customer in system: $T_{sys} = T_q + T_{ser}$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

A Little Queuing Theory

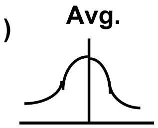


- Service time completions vs. waiting time for a busy server: randomly arriving event joins a queue of arbitrary length when server is busy, otherwise serviced immediately
  – Unlimited length queues key simplification
- A *single server queue*: combination of a servicing facility that accomodates 1 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?

A Little Queuing Theory

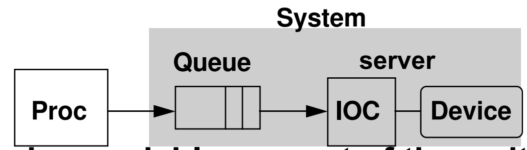


- **Server spends a variable amount of time with customers**
  - Weighted mean *m1* = (f1 x T1 + f2 x T2 +...+ fn x Tn)/F (F=f1 + f2...)
  - *variance* = (f1 x T1^2 + f2 x T2^2 +...+ fn x Tn^2)/F – m1^2
    - » Must keep track of unit of measure (100 ms^2 vs. 0.1 s^2)
  - *Squared coefficient of variance*: *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}

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

- **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 <u>M</u>arkovian request arrival (M for C=1 exponentially random), <u>G</u>eneral 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

CS 252 Administrivia

- Email URL of initial project home page to TA?
- Upcoming events in CS 252

18-Mar Wed I/O 3: Tertiary Storage & Network Intro
19-Mar Thu Send in Project Survey #2
20-Mar Fri Networks 2: Interface, Switches, Routing
23-Mar to 27-Mar Spring Break

A Little Queuing Theory: Another Example

- processor sends <u>20</u> x 8KB disk I/Os per sec, requests & service exponentially distrib., avg. disk service = <u>12 ms</u>
- 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 <u>20</u> x 8KB disk I/Os per sec, requests & service exponentially distrib., avg. disk service = <u>12 ms</u>
- 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 <u>10</u> x 8KB disk I/Os per second, <u>squared coef. var.(C) = 1.5</u>, avg. disk service time = <u>20 ms</u>**
- **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

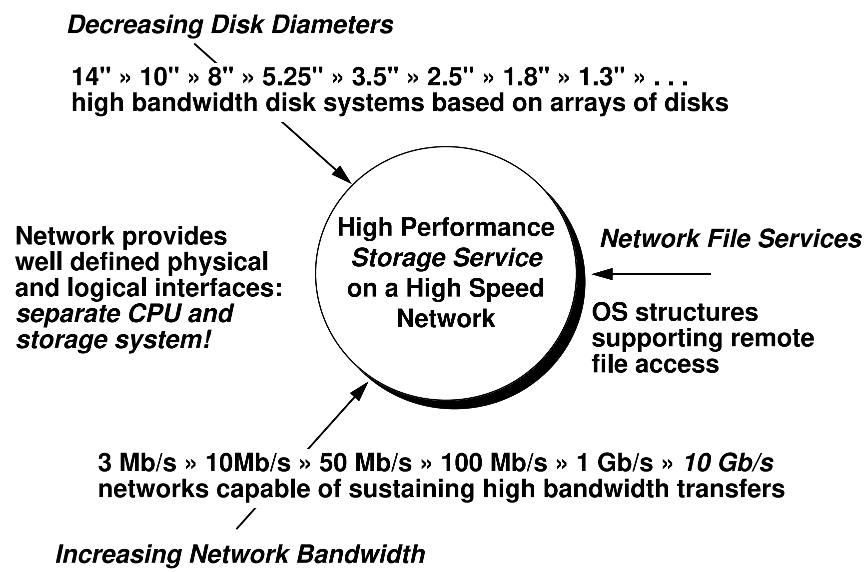
Pitfall of Not using Queuing Theory

- **1st 32-bit minicomputer (VAX-11/780)**
- **How big should write buffer be?**
  - Stores 10% of instructions, 1 MIPS
- **Buffer = 1**
- **=> Avg. Queue Length = 1 vs. low response time**

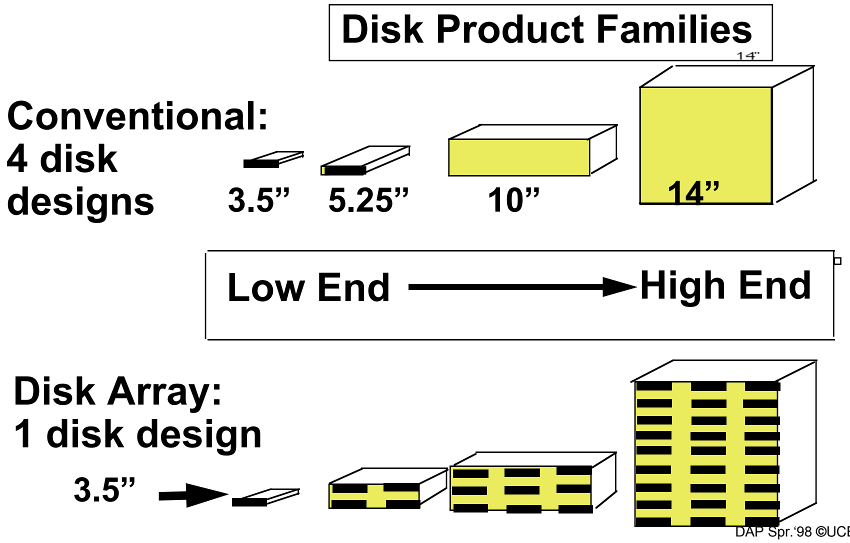
Review: Storage System Issues

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

Network Attached Storage



Manufacturing Advantages of Disk Arrays



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

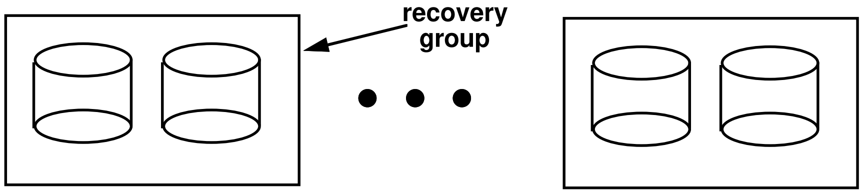
  Contents reconstructed from data redundantly stored in the array

  → 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*

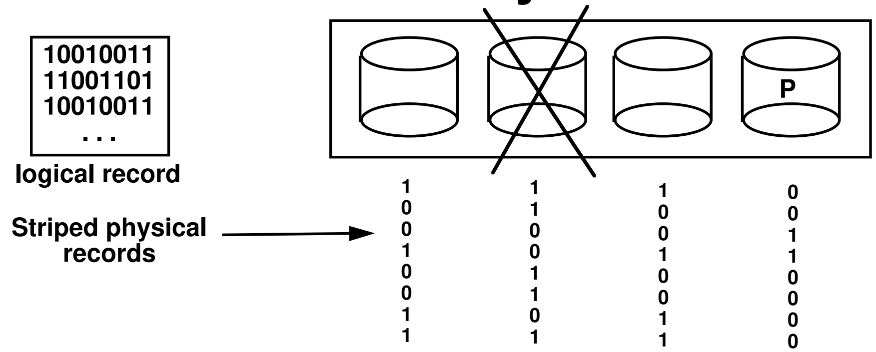
Redundant Arrays of Disks RAID 1: Disk Mirroring/Shadowing



- 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

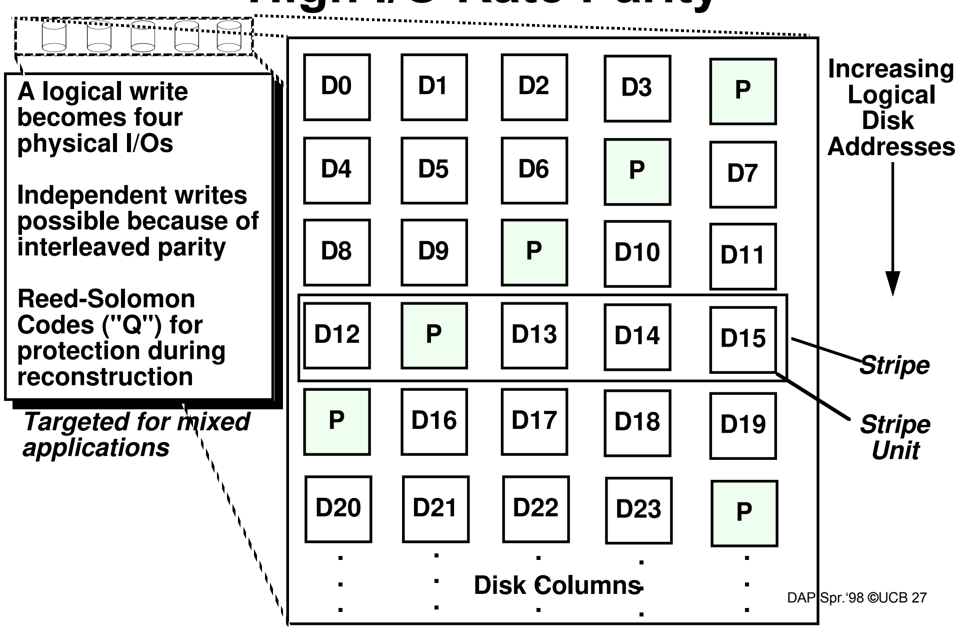
Redundant Arrays of Disks RAID 3: Parity Disk



- 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

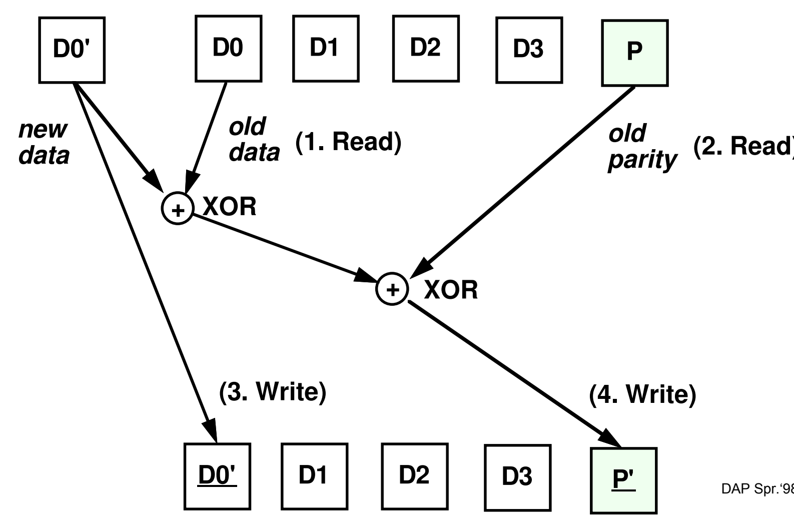
Redundant Arrays of Disks RAID 5+: High I/O Rate Parity

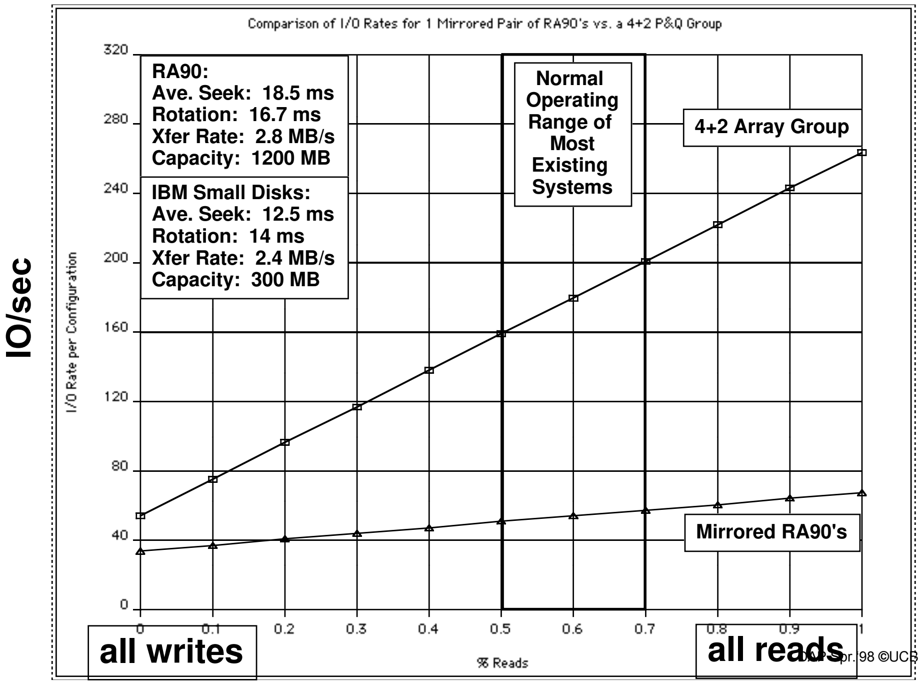


Problems of Disk Arrays: Small Writes

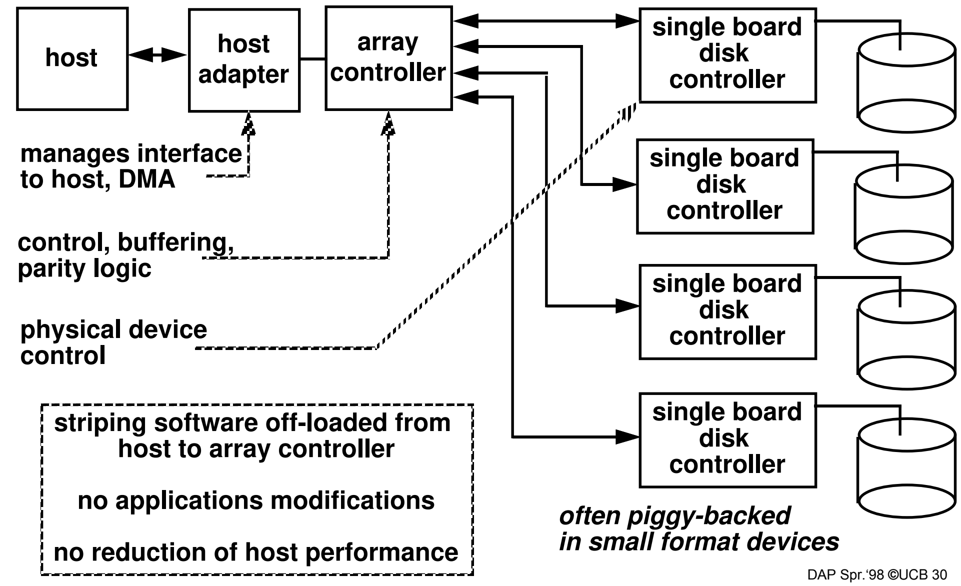
RAID-5: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes

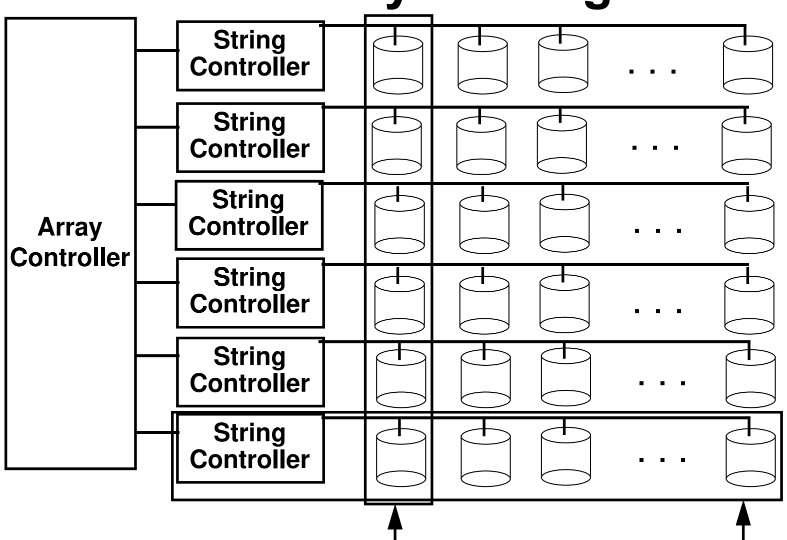




Subsystem Organization



System Availability: Orthogonal RAIDs

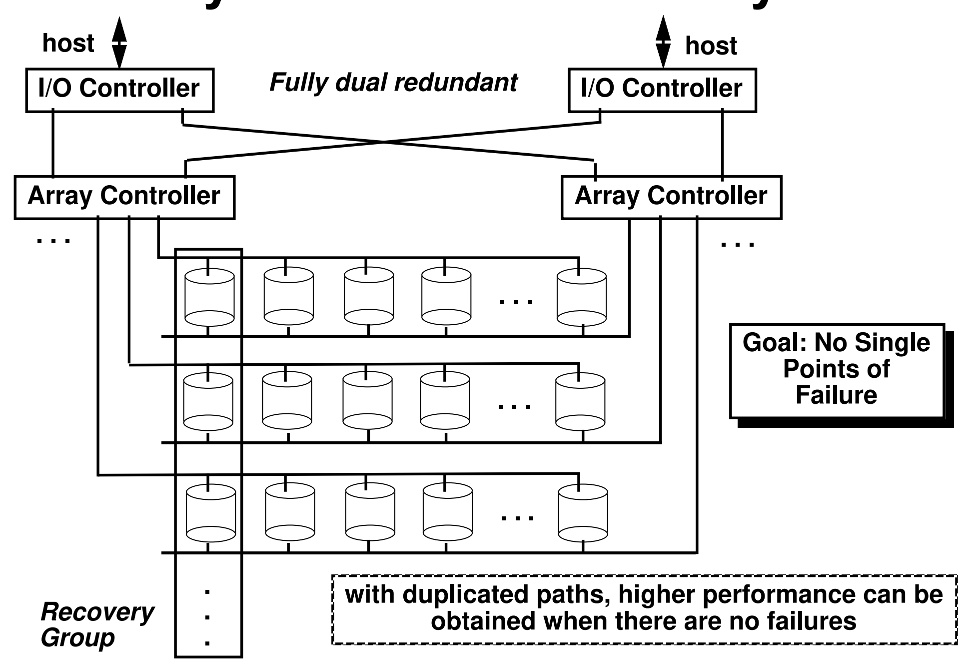


Data Recovery Group: **unit of data redundancy**

Redundant Support Components: **fans, power supplies, controller, cables**

End to End Data Integrity: **internal parity protected data paths**

System-Level Availability



Review: Storage System Issues

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

Interconnect Trends

- Interconnect = glue that interfaces computer system components
- High speed hardware interfaces + logical protocols
- Networks, channels, backplanes

	Network	Channel	Backplane
Distance	>1000 m	10 - 100 m	1 m
Bandwidth	10 - 100 Mb/s	40 - 1000 Mb/s	320 - 1000+ Mb/s
Latency	high (>ms)	medium	low (<µs)
Reliability	low Extensive CRC	medium Byte Parity	high Byte Parity

message-based
narrow pathways
distributed arb

⟷

memory-mapped
wide pathways
centralized arb

Backplane Architectures

Metric	VME	FutureBus	MultiBus II	SCSI-I
Bus Width (signals)	128	96	96	25
Address/Data Multiplexed?	No	Yes	Yes	na
Data Width	16 - 32	32	32	8
Xfer Size	Single/Multiple	Single/Multiple	Single/Multiple	Single/Multiple
# of Bus Masters	Multiple	Multiple	Multiple	Multiple
Split Transactions	No	Optional	Optional	Optional
Clocking	Async	Async	Sync	Either
Bandwidth, Single Word (0 ns mem)	25	37	20	5, 1.5
Bandwidth, Single Word (150 ns mem)	12.9	15.5	10	5, 1.5
Bandwidth Multiple Word (0 ns mem)	27.9	95.2	40	5, 1.5
Bandwidth Multiple Word (150 ns mem)	13.6	20.8	13.3	5, 1.5
Max # of devices	21	20	21	7
Max Bus Length	.5 m	.5 m	.5 m	25 m
Standard	IEEE 1014	IEEE 896	ANSI/IEEE 1296	ANSI X3.131

Distinctions begin to blur:

SCSI channel is like a bus

FutureBus is like a channel (disconnect/reconnect)

HIPPI forms links in high speed switching fabrics

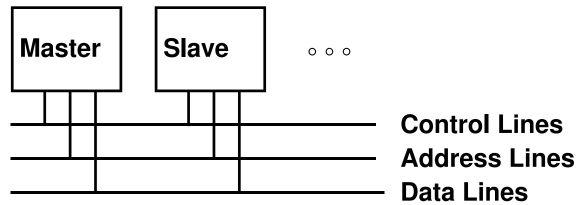
Bus-Based Interconnect

- Bus: a shared communication link between subsystems
  – Low cost: a single set of wires is shared multiple ways
  – Versatility: Easy to add new devices & peripherals may even be ported between computers using common bus
- Disadvantage
  – A communication bottleneck, possibly limiting the maximum I/O throughput
- Bus speed is limited by physical factors
  – the bus length
  – the number of devices (and, hence, bus loading).
  – these physical limits prevent arbitrary bus speedup.

Bus-Based Interconnect

- Two generic types of busses:
  - I/O busses: lengthy, many types of devices connected, wide range in the data bandwidth), and follow a bus standard (sometimes called a *channel*)
  - CPU–memory buses: high speed, matched to the memory system to maximize memory–CPU bandwidth, single device (sometimes called a *backplane*)
  - To lower costs, low cost (older) systems combine together
- Bus transaction
  - Sending address & receiving or sending data

Bus Protocols



Multibus: 20 address, 16 data, 5 control, 50ns Pause

Bus Master: has ability to control the bus, initiates transaction

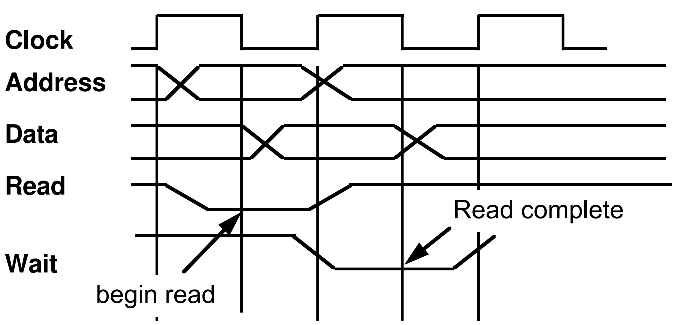
Bus Slave: module activated by the transaction

Bus Communication Protocol: specification of sequence of events and timing requirements in transferring information.

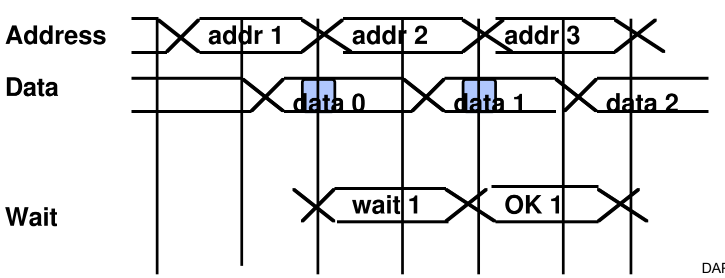
Asynchronous Bus Transfers: control lines (req., ack.) serve to orchestrate sequencing

Synchronous Bus Transfers: sequence relative to common clock

Synchronous Bus Protocols

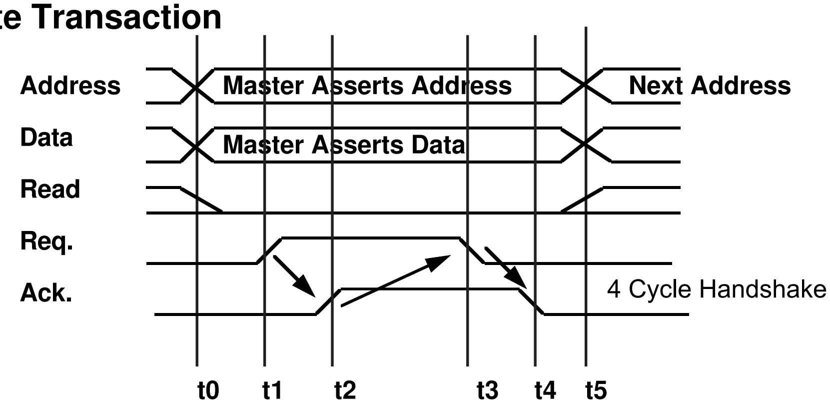


Pipelined/Split transaction Bus Protocol



Asynchronous Handshake

Write Transaction



t0 : Master has obtained control and asserts address, direction, data
Waits a specified amount of time for slaves to decode target\

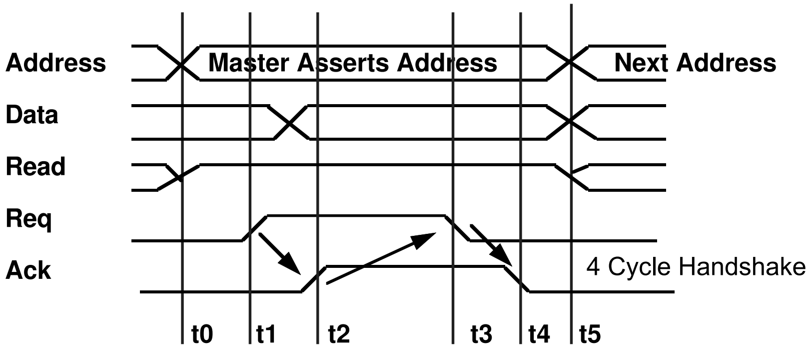
t1: Master asserts request line

t2: Slave asserts ack, indicating data received

t3: Master releases req

t4: Slave releases ack

Read Transaction



t0 : Master has obtained control and asserts address, direction, data
Waits a specified amount of time for slaves to decode target\

t1: Master asserts request line

t2: Slave asserts ack, indicating ready to transmit data

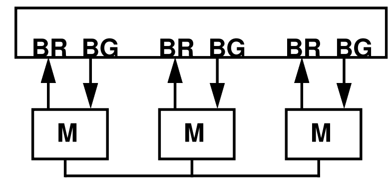
t3: Master releases req, data received

t4: Slave releases ack

Time Multiplexed Bus: address and data share lines

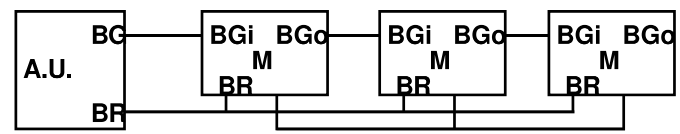
Bus Arbitration

Parallel (Centralized) Arbitration

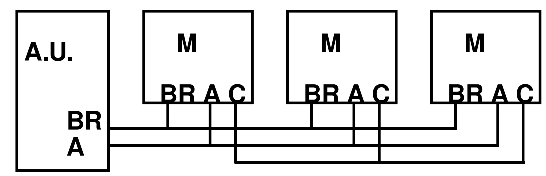


Bus Request
Bus Grant

Serial Arbitration (daisy chaining)



Polling



Bus Options

Option	*High performance*	*Low cost*
Bus width	Separate address & data lines	Multiplex address & data lines
Data width	Wider is faster (e.g., 32 bits)	Narrower is cheaper (e.g., 8 bits)
Transfer size	Multiple words has less bus overhead	Single-word transfer is simpler
Bus masters	Multiple (requires arbitration)	Single master (no arbitration)
Split transaction?	Yes—separate Request and Reply packets gets higher bandwidth (needs multiple masters)	No—continuous connection is cheaper and has lower latency
Clocking	Synchronous	Asynchronous

1990 Bus Survey (P&H, 1st Ed)

	VME	FutureBus	Multibus II	IPI	SCSI
Signals	128	96	96	16	8
Addr/Data mux	no	yes	yes	n/a	n/a
Data width	16 - 32	32	32	16	8
Masters	multi	multi	multi	single	multi
Clocking	Async	Async	Sync	Async	either
MB/s (0ns, word)	25	37	20	25	1.5 (asyn) 5 (sync)
150ns word	12.9	15.5	10	=	=
0ns block	27.9	95.2	40	=	=
150ns block	13.6	20.8	13.3	=	=
Max devices	21	20	21	8	7
Max meters	0.5	0.5	0.5	50	25
Standard	IEEE 1014	IEEE 896.1	ANSI/IEEE 1296	ANSI X3.129	ANSI X3.131

VME

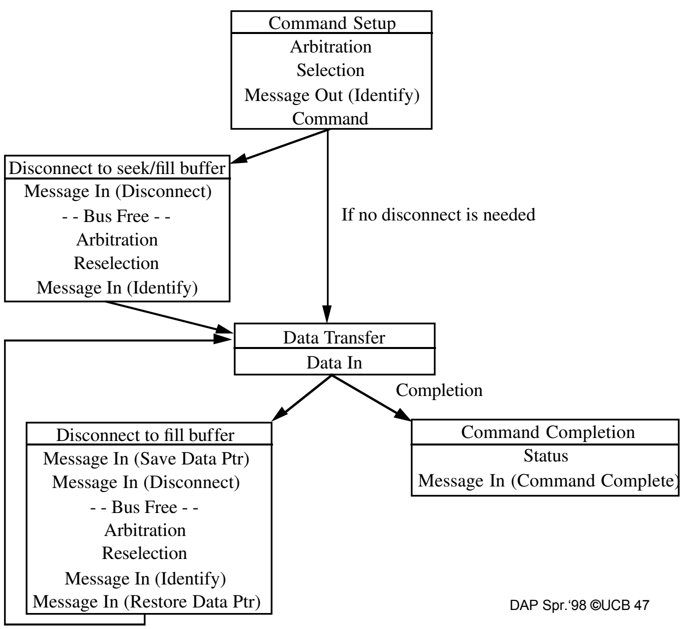
- 3 96-pin connectors
- 128 defined as standard, rest customer defined
  - 32 address
  - 32 data
  - 64 command & power/ground lines

SCSI: Small Computer System Interface

- Clock rate: 5 MHz / 10 MHz (fast) / 20 MHz (ultra)
- Width: n = 8 bits / 16 bits (wide); up to n – 1 devices to communicate on a bus or "string"
- Devices can be slave ("target") or master("initiator")
- SCSI protocol: a series of "phases", during which specific actions are taken by the controller and the SCSI disks
  - Bus Free: No device is currently accessing the bus
  - Arbitration: When the SCSI bus goes free, multiple devices may request (arbitrate for) the bus; fixed priority by address
  - Selection: informs the target that it will participate (Reselection if disconnected)
  - Command: the initiator reads the SCSI command bytes from host memory and sends them to the target
  - Data Transfer: data in or out, initiator: target
  - Message Phase: message in or out, initiator: target (identify, save/restore data pointer, disconnect, command complete)
  - Status Phase: target, just before command complete

SCSI "Bus": Channel Architecture

peer-to-peer protocols
initiator/target
linear byte streams
disconnect/reconnect



1993 I/O Bus Survey (P&H, 2nd Ed)

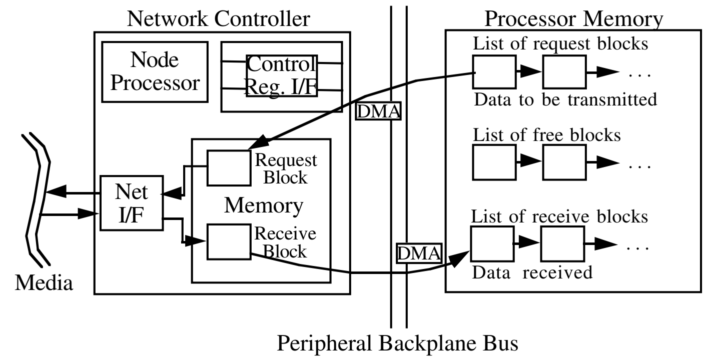
Bus	SBus	TurboChannel	MicroChannel	PCI
Originator	Sun	DEC	IBM	Intel
Clock Rate (MHz)	16-25	12.5-25	async	33
Addressing	Virtual	Physical	Physical	Physical
Data Sizes (bits)	8,16,32	8,16,24,32	8,16,24,32,64	8,16,24,32,64
Master	Multi	Single	Multi	Multi
Arbitration	Central	Central	Central	Central
32 bit read (MB/s)	33	25	20	33
Peak (MB/s)	89	84	75	111 (222)
Max Power (W)	16	26	13	25

1993 MP Server Memory Bus Survey

Bus	Summit	Challenge	XDBus
Originator	HP	SGI	Sun
Clock Rate (MHz)	60	48	66
Split transaction?	Yes	Yes	Yes?
Address lines	48	40	??
Data lines	128	256	144 (parity)
Data Sizes (bits)	512	1024	512
Clocks/transfer	4	5	4?
Peak (MB/s)	960	1200	1056
Master	Multi	Multi	Multi
Arbitration	Central	Central	Central
Addressing	Physical	Physical	Physical
Slots	16	9	10
Busses/system	1	1	2
Length	13 inches	12? inches	17 inches

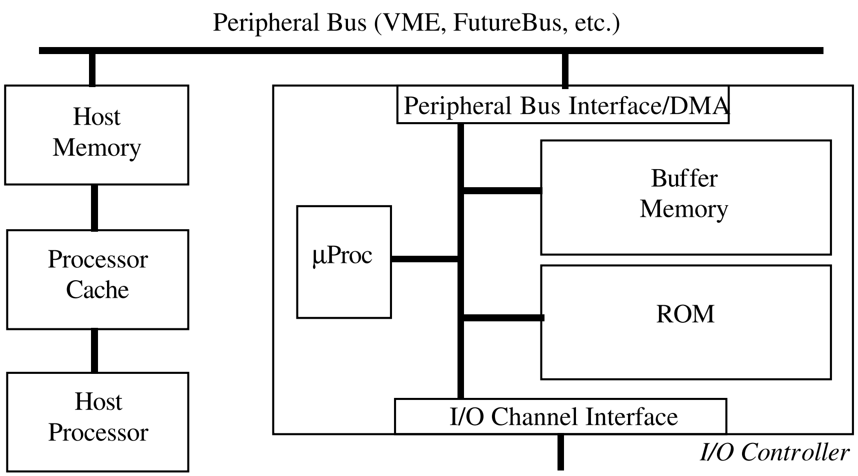
Communications Networks

Performance limiter is memory system, OS overhead



- **Send/receive queues in processor memories**
- **Network controller copies back and forth via DMA**
- **No host intervention needed**
- **Interrupt host when message sent or received**

I/O Controller Architecture

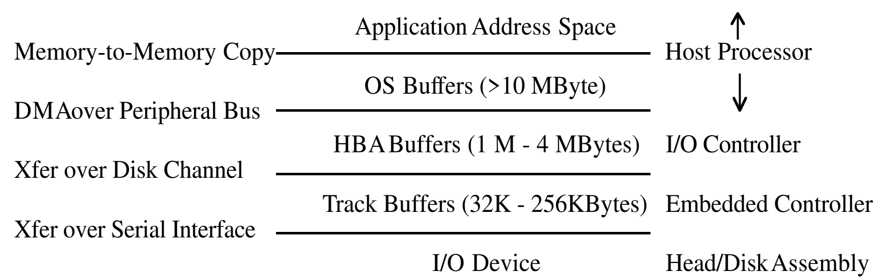


Request/response block interface

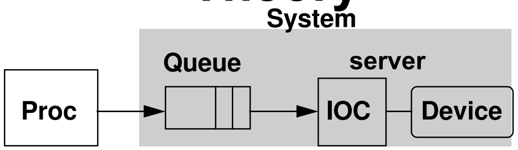
Backdoor access to host memory

I/O Data Flow

Impediment to high performance: multiple copies, complex hierarchy



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 x* $Time_{system}$
(Mean number customers = arrival rate x mean service time)

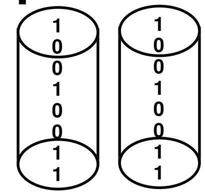
Summary: Redundant Arrays of Disks (RAID) Techniques

- *Disk Mirroring, Shadowing (RAID 1)*

Each disk is fully duplicated onto its "shadow"

Logical write = two physical writes

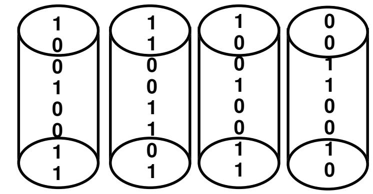
100% capacity overhead



- *Parity Data Bandwidth Array (RAID 3)*

Parity computed horizontally

Logically a single high data bw disk



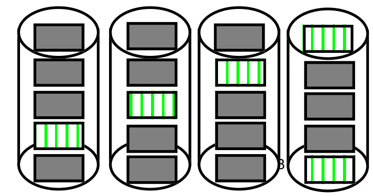
- *High I/O Rate Parity Array (RAID 5)*

Interleaved parity blocks

Independent reads and writes

Logical write = 2 reads + 2 writes

Parity + Reed-Solomon codes



Review: Storage System Issues

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