
EECS 252 Graduate Computer Architecture

Lec 18 – Storage

David Patterson
Electrical Engineering and Computer Sciences
University of California, Berkeley

<http://www.eecs.berkeley.edu/~pattsrn>
<http://vlsi.cs.berkeley.edu/cs252-s06>

Review

- **Virtual Machine Revival**
 - Overcome security flaws of modern OSes
 - Processor performance no longer highest priority
 - Manage Software, Manage Hardware
- **“... VMMs give OS developers another opportunity to develop functionality no longer practical in today’s complex and ossified operating systems, where innovation moves at geologic pace .”**
[Rosenblum and Garfinkel, 2005]
- **Virtualization challenges for processor, virtual memory, I/O**
 - Paravirtualization, ISA upgrades to cope with those difficulties
- **Xen as example VMM using paravirtualization**
 - 2005 performance on non-I/O bound, I/O intensive apps: 80% of native Linux without driver VM, 34% with driver VM
- **Opteron memory hierarchy still critical to performance**

4/12/2006

CS252 s06 Storage

2

Case for Storage

- **Shift in focus from computation to communication and storage of information**
 - E.g., Cray Research/Thinking Machines vs. Google/Yahoo
 - “The Computing Revolution” (1960s to 1980s)
⇒ “The Information Age” (1990 to today)
- **Storage emphasizes reliability and scalability as well as cost-performance**
- **What is “Software king” that determines which HW actually features used?**
 - Operating System for storage
 - Compiler for processor
- **Also has own performance theory—queuing theory—balances throughput vs. response time**

4/12/2006

CS252 s06 Storage

3

Outline

- **Magnetic Disks**
- **RAID**
- **Administrivia**
- **Advanced Dependability/Reliability/Availability**
- **I/O Benchmarks, Performance and Dependability**
- **Intro to Queuing Theory (if we have time)**
- **Conclusion**

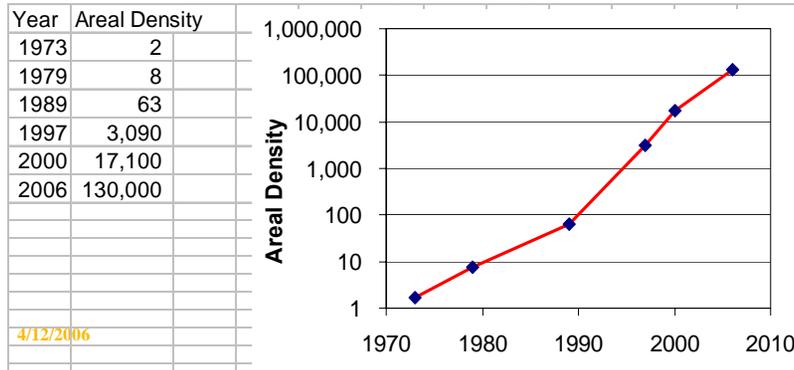
4/12/2006

CS252 s06 Storage

4

Disk Figure of Merit: Areal Density

- Bits recorded along a track
 - Metric is **Bits Per Inch (BPI)**
- Number of tracks per surface
 - Metric is **Tracks Per Inch (TPI)**
- Disk Designs Brag about **bit density per unit area**
 - Metric is **Bits Per Square Inch: Areal Density = BPI x TPI**



4/12/2006

Historical Perspective

- 1956 IBM Ramac — early 1970s Winchester
 - Developed for mainframe computers, proprietary interfaces
 - Steady shrink in form factor: 27 in. to 14 in.
- Form factor and capacity drives market more than performance
- 1970s developments
 - 5.25 inch floppy disk formfactor (microcode into mainframe)
 - Emergence of industry standard disk interfaces
- Early 1980s: PCs and first generation workstations
- Mid 1980s: Client/server computing
 - Centralized storage on file server
 - » accelerates disk downsizing: 8 inch to 5.25
 - Mass market disk drives become a reality
 - » industry standards: SCSI, IPI, IDE
 - » 5.25 inch to 3.5 inch drives for PCs, End of proprietary interfaces
- 1900s: Laptops => 2.5 inch drives
- 2000s: What new devices leading to new drives?

4/12/2006

CS252 s06 Storage

6

Future Disk Size and Performance

- Continued advance in capacity (60%/yr) and bandwidth (40%/yr)
- Slow improvement in seek, rotation (8%/yr)
- Time to read whole disk

Year	Sequentially	Randomly (1 sector/seek)
1990	4 minutes	6 hours
2000	12 minutes	1 week(!)
2006	56 minutes	3 weeks (SCSI)
2006	171 minutes	7 weeks (SATA)

4/12/2006

CS252 s06 Storage

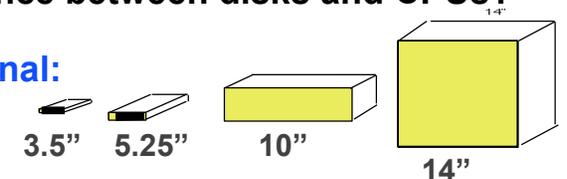
7

Use Arrays of Small Disks?

•Katz and Patterson asked in 1987:

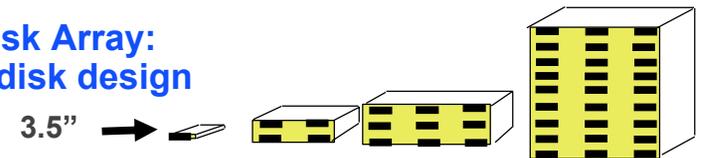
- Can smaller disks be used to close gap in performance between disks and CPUs?

Conventional:
4 disk designs



Low End → High End

Disk Array:
1 disk design

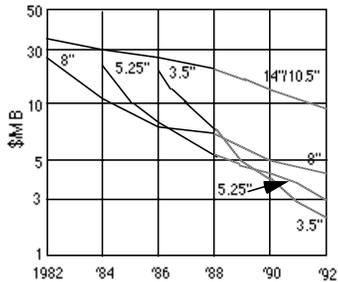


4/12/2006

CS252 s06 Storage

8

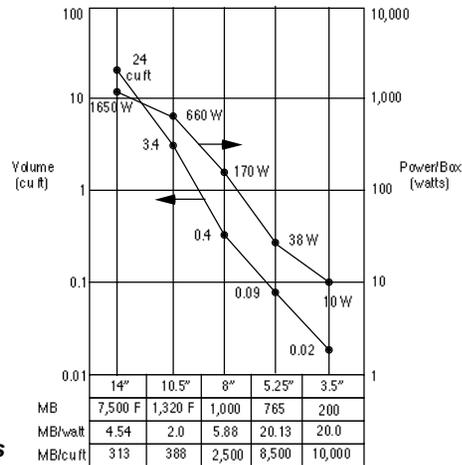
Advantages of Small Formfactor Disk Drives



Low cost/MB
High MB/volume
High MB/watt
Low cost/Actuator

Cost and Environmental Efficiencies

4/12/2006



CS252 s06 Storage

9

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

	IBM 3390K	IBM 3.5" 0061	x70
Capacity	20 GBytes	320 MBytes	23 GBytes
Volume	97 cu. ft.	0.1 cu. ft.	11 cu. ft. 9X
Power	3 KW	11 W	1 KW 3X
Data Rate	15 MB/s	1.5 MB/s	120 MB/s 8X
I/O Rate	600 I/Os/s	55 I/Os/s	3900 IOs/s 6X
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, but what about reliability?

CS252 s06 Storage

10

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

4/12/2006

CS252 s06 Storage

11

Redundant Arrays of (Inexpensive) Disks

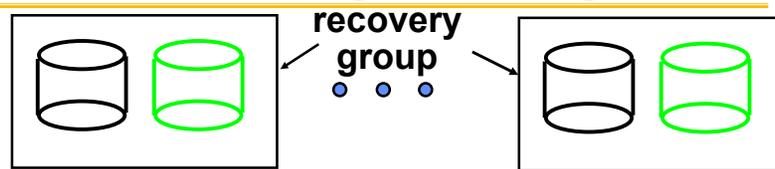
- Files are "striped" across multiple disks
- Redundancy yields high data availability
 - **Availability**: service still provided to user, even if some components failed
- Disks will still fail
- Contents reconstructed from data redundantly stored in the array
 - ⇒ Capacity penalty to store redundant info
 - ⇒ Bandwidth penalty to update redundant info

4/12/2006

CS252 s06 Storage

12

Redundant Arrays of Inexpensive Disks RAID 1: Disk Mirroring/Shadowing



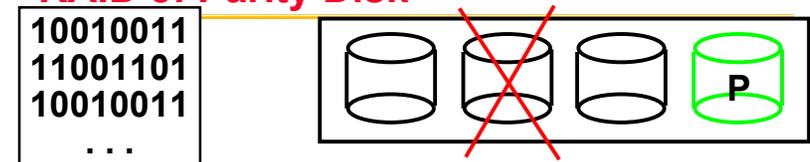
- Each disk is fully duplicated onto its “**mirror**”
Very high availability can be achieved
- Bandwidth sacrifice on write:
Logical write = two physical writes
 - Reads may be optimized
- Most expensive solution: 100% capacity overhead
- (RAID 2 not interesting, so skip)

4/12/2006

CS252 s06 Storage

13

Redundant Array of Inexpensive Disks RAID 3: Parity Disk



logical record	1	1	1	1
	0	1	0	1
Striped physical records	1	0	1	0
	0	0	0	0
P contains sum of other disks per stripe mod 2 (“ parity ”)	0	1	0	1
	0	1	0	1
If disk fails, subtract P from sum of other disks to find missing information	1	0	1	0
	1	1	1	1

4/12/2006

CS252 s06 Storage

14

RAID 3

- Sum computed across recovery group to protect against hard disk failures, stored in P disk
- Logically, a single high capacity, high transfer rate disk: good for large transfers
- Wider arrays reduce capacity costs, but decreases availability
- 33% capacity cost for parity if 3 data disks and 1 parity disk

4/12/2006

CS252 s06 Storage

15

Inspiration for RAID 4

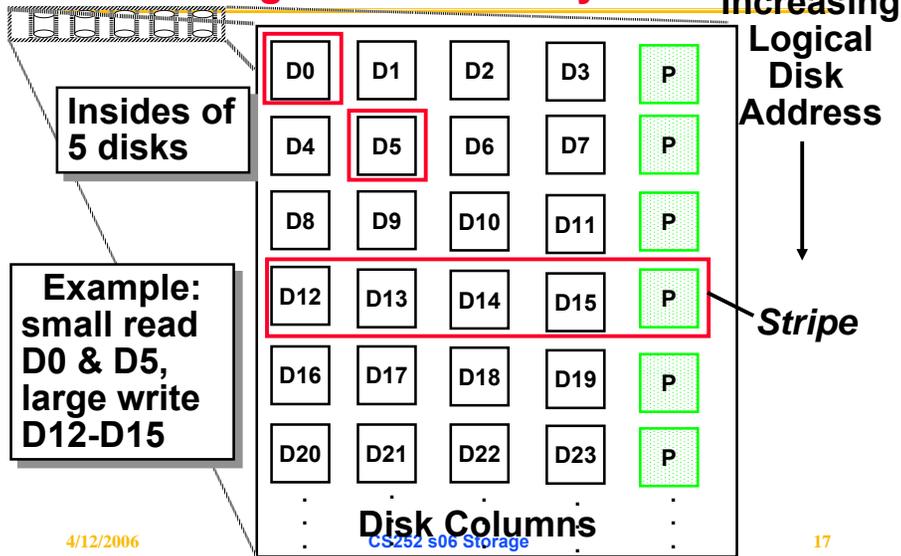
- RAID 3 relies on parity disk to discover errors on Read
- But every sector has an error detection field
- To catch errors on read, rely on error detection field vs. the parity disk
- Allows independent reads to different disks simultaneously

4/12/2006

CS252 s06 Storage

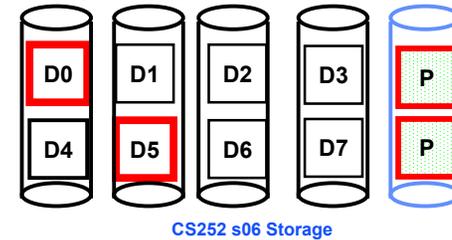
16

Redundant Arrays of Inexpensive Disks RAID 4: High I/O Rate Parity

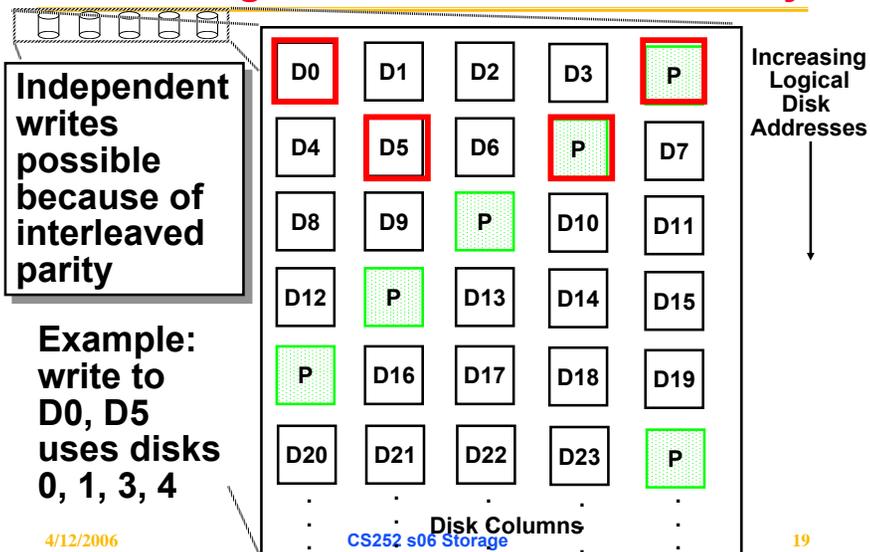


Inspiration for RAID 5

- RAID 4 works well for small reads
- Small writes (write to one disk):
 - Option 1: read other data disks, create new sum and write to Parity Disk
 - Option 2: since P has old sum, compare old data to new data, add the difference to P
- Small writes are limited by Parity Disk: Write to D0, D5 both also write to P disk



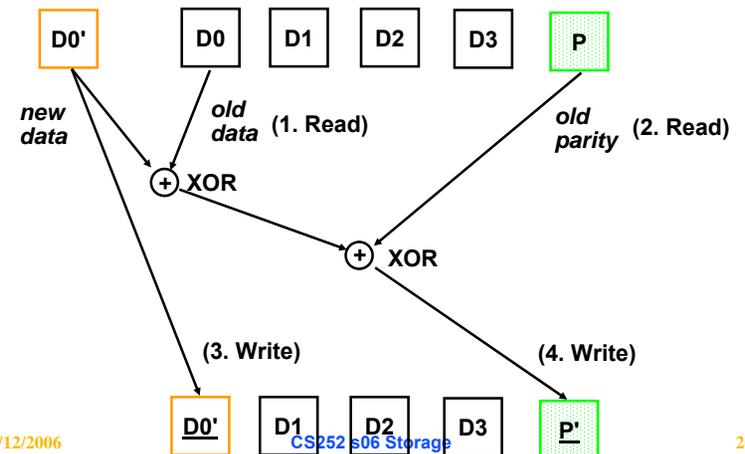
Redundant Arrays of Inexpensive Disks RAID 5: High I/O Rate Interleaved Parity



Problems of Disk Arrays: Small Writes

RAID-5: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes



CS252: Administrivia

- Wed 4/12 – Mon 4/17 Storage (Ch 6)
- RAMP Blue meeting Today 3:30-4 380 Soda
- Makeup Pizza: LaVal's on Euclid, 6-7 PM
- Project Update Meeting Wednesday 4/19
- Monday 4/24 Quiz 2 5-8 PM (Mainly Ch 4 to 6)
- Wed 4/26 Bad Career Advice / Bad Talk Advice
- Project Presentations Monday 5/1 (all day)
- Project Posters 5/3 Wednesday (11-1 in Soda)
- Final Papers due Friday 5/5 (email Archana, who will post papers on class web site)

4/12/2006

CS252 s06 Storage

21

CS252: Administrivia

- Fri 4/14 Read, comment RAID Paper and Homework. Be sure to answer
 - What was main motivation for RAID in paper?
 - Did prediction of processor performance and disk capacity hold?
 - How propose balance performance and capacity of RAID 1 to RAID 5? What do you think of it?
 - What were some of the open issues? Which were significant
 - In retrospect, what do you think were important contributions? What did the authors get wrong?

4/12/2006

CS252 s06 Storage

22

RAID 6: Recovering from 2 failures

- Why > 1 failure recovery?
 - operator accidentally replaces the wrong disk during a failure
 - since disk bandwidth is growing more slowly than disk capacity, the MTT Repair a disk in a RAID system is increasing
 - ⇒ increases the chances of a 2nd failure during repair since takes longer
 - reading much more data during reconstruction meant increasing the chance of an uncorrectable media failure, which would result in data loss

4/12/2006

CS252 s06 Storage

23

RAID 6: Recovering from 2 failures

- Network Appliance's *row-diagonal parity* or *RAID-DP*
- Like the standard RAID schemes, it uses redundant space based on parity calculation per stripe
- Since it is protecting against a double failure, it adds two check blocks per stripe of data.
 - If $p+1$ disks total, $p-1$ disks have data; assume $p=5$
- Row parity disk is just like in RAID 4
 - Even parity across the other 4 data blocks in its stripe
- Each block of the diagonal parity disk contains the even parity of the blocks in the same diagonal

4/12/2006

CS252 s06 Storage

24

Example p = 5

- Row diagonal parity starts by recovering one of the 4 blocks on the failed disk using diagonal parity
 - Since each diagonal misses one disk, and all diagonals miss a different disk, 2 diagonals are only missing 1 block
- Once the data for those blocks is recovered, then the standard RAID recovery scheme can be used to recover two more blocks in the standard RAID 4 stripes
- Process continues until two failed disks are restored

Data Disk 0	Data Disk 1	Data Disk 2	Data Disk 3	Row Parity	Diagonal Parity
0	1	2	3	4	0
1	2	3	4	0	1
2	3	4	0	1	2
3	4	0	1	2	3
4	0	1	2	3	4
0	1	2	3	4	0

4/12/2006

CS252 s06 Storage

25

Berkeley History: RAID-I

- **RAID-I (1989)**
 - Consisted of a Sun 4/280 workstation with 128 MB of DRAM, four dual-string SCSI controllers, 28 5.25-inch SCSI disks and specialized disk striping software
- Today RAID is \$24 billion dollar industry, 80% nonPC disks sold in RAID5



4/12/2006

CS252 s06 Storage

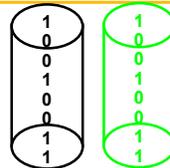
Summary: RAID Techniques: Goal was performance, popularity due to reliability of storage

• Disk Mirroring, Shadowing (RAID 1)

Each disk is fully duplicated onto its "shadow"

Logical write = two physical writes

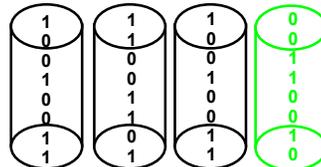
100% capacity overhead



• Parity Data Bandwidth Array (RAID 3)

Parity computed horizontally

Logically a single high data bw disk

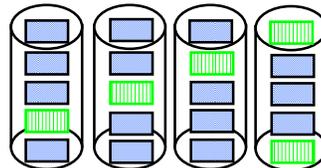


• High I/O Rate Parity Array (RAID 5)

Interleaved parity blocks

Independent reads and writes

Logical write = 2 reads + 2 writes



4/12/2006

CS252 s06 Storage

Definitions

- **Examples on why precise definitions so important for reliability**
- Is a programming mistake a fault, error, or failure?
 - Are we talking about the time it was designed or the time the program is run?
 - If the running program doesn't exercise the mistake, is it still a fault/error/failure?
- If an alpha particle hits a DRAM memory cell, is it a fault/error/failure if it doesn't change the value?
 - Is it a fault/error/failure if the memory doesn't access the changed bit?
 - Did a fault/error/failure still occur if the memory had error correction and delivered the corrected value to the CPU?

4/12/2006

CS252 s06 Storage

28

IFIP Standard terminology

- Computer system **dependability**: quality of delivered service such that reliance can be placed on service
- **Service** is observed **actual behavior** as perceived by other system(s) interacting with this system's users
- Each module has ideal **specified behavior**, where **service specification** is agreed description of expected behavior
- A system **failure** occurs when the actual behavior deviates from the specified behavior
- failure occurred because an **error**, a defect in module
- The cause of an error is a **fault**
- When a fault occurs it creates a **latent error**, which becomes **effective** when it is activated
- When error actually affects the delivered service, a failure occurs (time from error to failure is **error latency**)

4/12/2006

CS252 s06 Storage

29

Fault v. (Latent) Error v. Failure

- An **error** is manifestation *in the system* of a **fault**, a **failure** is manifestation *on the service* of an **error**
- Is If an alpha particle hits a DRAM memory cell, is it a fault/error/failure if it doesn't change the value?
 - Is it a fault/error/failure if the memory doesn't access the changed bit?
 - Did a fault/error/failure still occur if the memory had error correction and delivered the corrected value to the CPU?
- An alpha particle hitting a DRAM can be a **fault**
- if it changes the memory, it creates an **error**
- error remains **latent** until effected memory word is read
- if the effected word error affects the delivered service, a **failure** occurs

4/12/2006

CS252 s06 Storage

30

Fault Categories

1. **Hardware faults**: Devices that fail, such alpha particle hitting a memory cell
2. **Design faults**: Faults in software (usually) and hardware design (occasionally)
3. **Operation faults**: Mistakes by operations and maintenance personnel
4. **Environmental faults**: Fire, flood, earthquake, power failure, and sabotage
 - Also by duration:
 1. **Transient faults** exist for limited time and not recurring
 2. **Intermittent faults** cause a system to oscillate between faulty and fault-free operation
 3. **Permanent faults** do not correct themselves over time

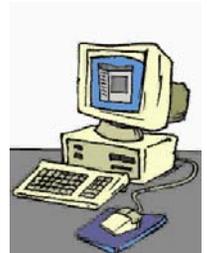
4/12/2006

CS252 s06 Storage

31

Fault Tolerance vs Disaster Tolerance

- Fault-Tolerance (or more properly, Error-Tolerance): **mask local faults (prevent errors from becoming failures)**
 - RAID disks
 - Uninterruptible Power Supplies
 - Cluster Failover
- Disaster Tolerance: **masks site errors (prevent site errors from causing service failures)**
 - Protects against fire, flood, sabotage,..
 - Redundant system and service at remote site.
 - Use design diversity



From Jim Gray's "Talk at UC Berkeley on Fault Tolerance" 11/9/00

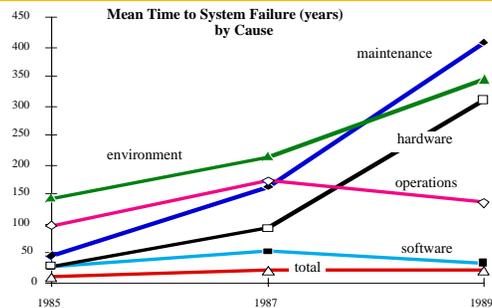
4/12/2006

CS252 s06 Storage

32

Case Studies - Tandem Trends

Reported MTTF by Component



	1985	1987	1990	
SOFTWARE	2	53	33	Years
HARDWARE	29	91	310	Years
MAINTENANCE	45	162	409	Years
OPERATIONS	99	171	136	Years
ENVIRONMENT	142	214	346	Years
SYSTEM	8	20	21	Years

Problem: Systematic Under-reporting

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance" 11/9/00

4/12/2006

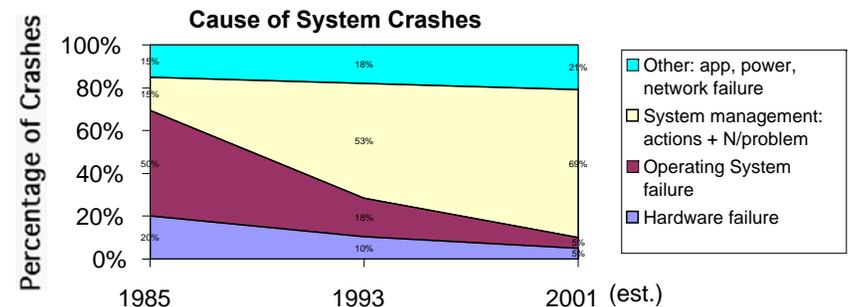
CS252 s06 Storage

33

Is Maintenance the Key?

• Rule of Thumb: Maintenance 10X HW

– so over 5 year product life, ~ 95% of cost is maintenance



• VAX crashes '85, '93 [Murp95]; extrap. to '01

• Sys. Man.: N crashes/problem, SysAdmin action

– Actions: set params bad, bad config, bad app install

• HW/OS 70% in '85 to 28% in '93. In '01, 10%?

4/12/2006

CS252 s06 Storage

34

HW Failures in Real Systems: Tertiary Disks

• A cluster of 20 PCs in seven 7-foot high, 19-inch wide racks with 368 8.4 GB, 7200 RPM, 3.5-inch IBM disks. The PCs are P6-200MHz with 96 MB of DRAM each. They run FreeBSD 3.0 and the hosts are connected via switched 100 Mbit/second Ethernet

Component	Total in System	Total Failed	% Failed
SCSI Controller	44	1	2.3%
SCSI Cable	39	1	2.6%
SCSI Disk	368	7	1.9%
IDE Disk	24	6	25.0%
Disk Enclosure -Backplane	46	13	28.3%
Disk Enclosure - Power Supply	92	3	3.3%
Ethernet Controller	20	1	5.0%
Ethernet Switch	2	1	50.0%
Ethernet Cable	42	1	2.3%
CPU/Motherboard	20	0	0%

4/12/2006

CS252 s06 Storage

35

Does Hardware Fail Fast? 4 of 384 Disks that failed in Tertiary Disk

Messages in system log for failed disk	No. log msgs	Duration (hours)
Hardware Failure (Peripheral device write fault [for] Field Replaceable Unit)	1763	186
Not Ready (Diagnostic failure: ASCQ = Component ID [of] Field Replaceable Unit)	1460	90
Recovered Error (Failure Prediction Threshold Exceeded [for] Field Replaceable Unit)	1313	5
Recovered Error (Failure Prediction Threshold Exceeded [for] Field Replaceable Unit)	431	17

4/12/2006

CS252 s06 Storage

36

High Availability System Classes Goal: Build Class 6 Systems

System Type	Unavailable (min/year)	Availability	Availability Class
Unmanaged	50,000	90.0%	1
Managed	5,000	99.0%	2
Well Managed	500	99.9%	3
Fault Tolerant	50	99.99%	4
High-Availability	5	99.999%	5
Very-High-Availability	.5	99.9999%	6
Ultra-Availability	.05	99.99999%	7

UnAvailability = MTTR/MTBF
can cut it in 1/2 by cutting MTTR *or* MTBF

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance" 11/9/00
4/12/2006 CS252 s06 Storage

37

How Realistic is "5 Nines"?

- HP claims HP-9000 server HW and HP-UX OS can deliver 99.999% availability guarantee "in certain pre-defined, pre-tested customer environments"
 - Application faults?
 - Operator faults?
 - Environmental faults?
- Collocation sites (lots of computers in 1 building on Internet) have
 - 1 network outage per year (~1 day)
 - 1 power failure per year (~1 day)
- Microsoft Network unavailable recently for a day due to problem in Domain Name Server: if only outage per year, 99.7% or 2 Nines

4/12/2006

CS252 s06 Storage

38

Outline

- Magnetic Disks
- RAID
- Administrivia
- Advanced Dependability/Reliability/Availability
- I/O Benchmarks, Performance and Dependability
- Intro to Queueing Theory (if we have time)
- Conclusion

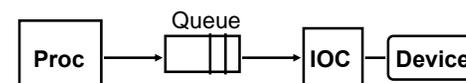
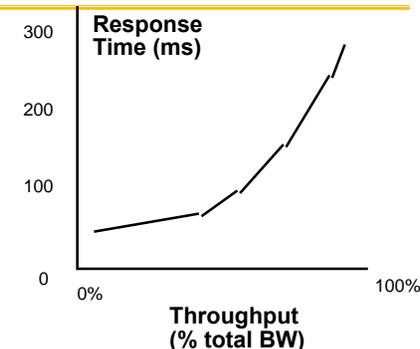
4/12/2006

CS252 s06 Storage

39

I/O Performance

Metrics:
Response Time vs. Throughput



Response time = Queue + Device Service time

4/12/2006

CS252 s06 Storage

40

I/O Benchmarks

- For better or worse, benchmarks shape a field
 - Processor benchmarks classically aimed at response time for fixed sized problem
 - I/O benchmarks typically measure throughput, possibly with upper limit on response times (or 90% of response times)
- Transaction Processing (TP) (or On-line TP=OLTP)
 - If bank computer fails when customer withdraw money, TP system guarantees account debited if customer gets \$ & account unchanged if no \$
 - Airline reservation systems & banks use TP
- Atomic transactions makes this work
- Classic metric is Transactions Per Second (TPS)

4/12/2006

CS252 s06 Storage

41

I/O Benchmarks: Transaction Processing

- Early 1980s great interest in OLTP
 - Expecting demand for high TPS (e.g., ATM machines, credit cards)
 - Tandem's success implied medium range OLTP expands
 - Each vendor picked own conditions for TPS claims, report only CPU times with widely different I/O
 - Conflicting claims led to disbelief of all benchmarks ⇒ chaos
- 1984 Jim Gray (Tandem) distributed paper to Tandem + 19 in other companies propose standard benchmark
- Published "A measure of transaction processing power," Datamation, 1985 by Anonymous et. al
 - To indicate that this was effort of large group
 - To avoid delays of legal department of each author's firm
 - Still get mail at Tandem to author "Anonymous"
- Led to Transaction Processing Council in 1988
 - www.tpc.org

4/12/2006

CS252 s06 Storage

42

I/O Benchmarks: TP1 by Anon et. al

- DebitCredit Scalability: size of account, branch, teller, history function of throughput

TPS	Number of ATMs	Account-file size
10	1,000	0.1 GB
100	10,000	1.0 GB
1,000	100,000	10.0 GB
10,000	1,000,000	100.0 GB

 - Each input TPS =>100,000 account records, 10 branches, 100 ATMs
 - Accounts must grow since a person is not likely to use the bank more frequently just because the bank has a faster computer!
- Response time: 95% transactions take ≤ 1 second
- Report price (initial purchase price + 5 year maintenance = cost of ownership)
- Hire auditor to certify results

4/12/2006

CS252 s06 Storage

43

Unusual Characteristics of TPC

- Price is included in the benchmarks
 - cost of HW, SW, and 5-year maintenance agreements included ⇒ price-performance as well as performance
- The data set generally must scale in size as the throughput increases
 - trying to model real systems, demand on system and size of the data stored in it increase together
- The benchmark results are audited
 - Must be approved by certified TPC auditor, who enforces TPC rules ⇒ only fair results are submitted
- Throughput is the performance metric but response times are limited
 - eg, TPC-C: 90% transaction response times < 5 seconds
- An independent organization maintains the benchmarks

4/12/2006

CS252 s06 Storage

44

– COO ballots on changes, meetings, to settle disputes...

TPC Benchmark History/Status

Benchmark	Data Size (GB)	Performance Metric	1st Results
A: Debit Credit (retired)	0.1 to 10	transactions/s	Jul-90
B: Batch Debit Credit (retired)	0.1 to 10	transactions/s	Jul-91
C: Complex Query OLTP	100 to 3000 (min. 07 * tpm)	new order trans/min (tpm)	Sep-92
D: Decision Support (retired)	100, 300, 1000	queries/hour	Dec-95
H: Ad hoc decision support	100, 300, 1000	queries/hour	Oct-99
R: Business reporting decision support (retired)	1000	queries/hour	Aug-99
W: Transactional web	~ 50, 500	web interactions/sec.	Jul-00
App: app. server & web services		Web Service Interactions/sec (SIPS)	Jun-05

4/12/2006

CS252 s06 Storage

45

I/O Benchmarks via SPEC

- **SFS 3.0 Attempt by NFS companies to agree on standard benchmark**
 - Run on multiple clients & networks (to prevent bottlenecks)
 - Same caching policy in all clients
 - Reads: 85% full block & 15% partial blocks
 - Writes: 50% full block & 50% partial blocks
 - Average response time: 40 ms
 - Scaling: for every 100 NFS ops/sec, increase capacity 1GB
- **Results: plot of server load (throughput) vs. response time & number of users**
 - Assumes: 1 user => 10 NFS ops/sec
 - 3.0 for NSF 3.0
- **Added SPECmail (mailserver), SPECweb (webserver) benchmarks**

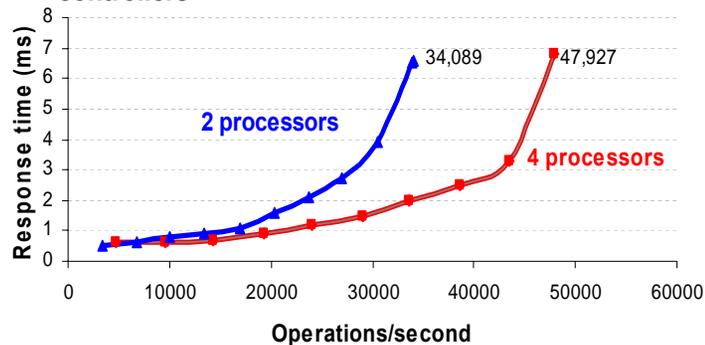
4/12/2006

CS252 s06 Storage

46

2005 Example SPEC SFS Result: NetApp FAS3050c NFS servers

- 2.8 GHz Pentium Xeon microprocessors, 2 GB of DRAM per processor, 1GB of Non-volatile memory per system
- 4 FDDI networks; 32 NFS Daemons, 24 GB file size
- 168 fibre channel disks: 72 GB, 15000 RPM, 2 or 4 FC controllers



4/12/2006

CS252 s06 Storage

47

Availability benchmark methodology

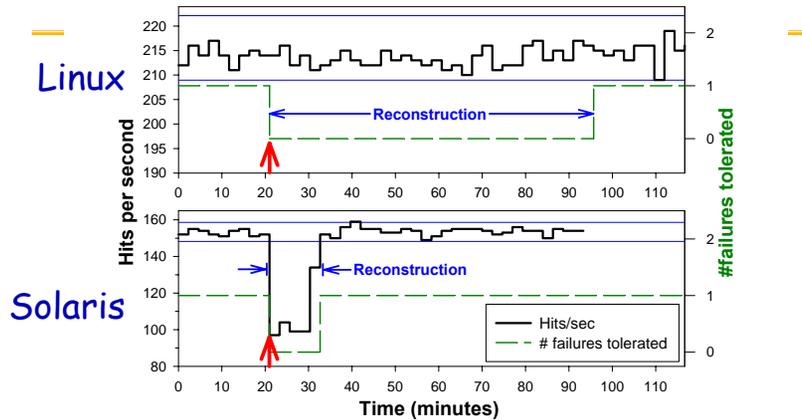
- **Goal: quantify variation in QoS metrics as events occur that affect system availability**
- **Leverage existing performance benchmarks**
 - to generate fair workloads
 - to measure & trace quality of service metrics
- **Use fault injection to compromise system**
 - hardware faults (disk, memory, network, power)
 - software faults (corrupt input, driver error returns)
 - maintenance events (repairs, SW/HW upgrades)
- **Examine *single-fault* and *multi-fault* workloads**
 - the availability analogues of performance micro- and macro-benchmarks

4/12/2006

CS252 s06 Storage

48

Example single-fault result



- Compares Linux and Solaris reconstruction
 - Linux: **minimal performance impact but longer window of vulnerability to second fault**
 - Solaris: **large perf. impact but restores redundancy fast**

4/12/2006

CS252 s06 Storage

49

Reconstruction policy (2)

- **Linux:** favors performance over data availability
 - automatically-initiated reconstruction, idle bandwidth
 - virtually no performance impact on application
 - very long window of vulnerability (>1hr for 3GB RAID)
- **Solaris:** favors data availability over app. perf.
 - automatically-initiated reconstruction at high BW
 - as much as 34% drop in application performance
 - short window of vulnerability (10 minutes for 3GB)
- **Windows:** favors neither!
 - *manually-initiated* reconstruction at moderate BW
 - as much as 18% app. performance drop
 - somewhat short window of vulnerability (23 min/3GB)

4/12/2006

CS252 s06 Storage

50

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 black box not creating or destroying tasks

4/12/2006

CS252 s06 Storage

51

Deriving Little's Law

- $\text{Time}_{\text{observe}}$ = elapsed time that observe a system
 - $\text{Number}_{\text{task}}$ = number of task during $\text{Time}_{\text{observe}}$
 - $\text{Time}_{\text{accumulated}}$ = sum of elapsed times for each task
- Then
- **Mean number tasks in system** = $\text{Time}_{\text{accumulated}} / \text{Time}_{\text{observe}}$
 - **Mean response time** = $\text{Time}_{\text{accumulated}} / \text{Number}_{\text{task}}$
 - **Arrival Rate** = $\text{Number}_{\text{task}} / \text{Time}_{\text{observe}}$
- Factoring RHS of 1st equation
- $\text{Time}_{\text{accumulated}} / \text{Time}_{\text{observe}} = \text{Time}_{\text{accumulated}} / \text{Number}_{\text{task}} \times \text{Number}_{\text{task}} / \text{Time}_{\text{observe}}$

Then get Little's Law:

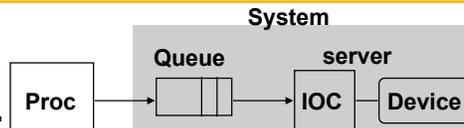
- **Mean number tasks in system** = **Arrival Rate** x **Mean response time**

4/12/2006

CS252 s06 Storage

52

A Little Queuing Theory: Notation



- **Notation:**
 - $Time_{server}$ average time to service a task
 - Average service rate = $1 / Time_{server}$ (traditionally μ)
 - $Time_{queue}$ average time/task in queue
 - $Time_{system}$ average time/task in system
= $Time_{queue} + Time_{server}$
 - Arrival rate avg no. of arriving tasks/sec (traditionally λ)
 - $Length_{server}$ average number of tasks in service
 - $Length_{queue}$ average length of queue
 - $Length_{system}$ average number of tasks in service
= $Length_{queue} + Length_{server}$
 - Little's Law: $Length_{server} = Arrival\ rate \times Time_{server}$
(Mean number tasks = arrival rate x mean service time)

4/12/2006

CS252 s06 Storage

53

Server Utilization

- For a single server, service rate = $1 / Time_{server}$
- **Server utilization** must be between 0 and 1, since system is in equilibrium (arrivals = departures); often called **traffic intensity**, traditionally ρ
- **Server utilization = mean number tasks in service = Arrival rate x Timeserver**
- What is disk utilization if get 50 I/O requests per second for disk and average disk service time is 10 ms (0.01 sec)?
- Server utilization = $50/\text{sec} \times 0.01 \text{ sec} = 0.5$
- Or server is busy on average 50% of time

4/12/2006

CS252 s06 Storage

54

Time in Queue vs. Length of Queue

- We assume First In First Out (FIFO) queue
- Relationship of time in queue ($Time_{queue}$) to mean number of tasks in queue ($Length_{queue}$) ?
- $Time_{queue} = Length_{queue} \times Time_{server}$
+ "Mean time to complete service of task when new task arrives if server is busy"
- New task can arrive at any instant; how predict last part?
- To predict performance, need to know sometime about distribution of events

4/12/2006

CS252 s06 Storage

55

Poisson Distribution of Random Variables

- A variable is random if it takes one of a specified set of values with a specified probability
 - you cannot know exactly what its next value will be, but you may know the probability of all possible values
- I/O Requests can be modeled by a random variable because OS normally switching between several processes generating independent I/O requests
 - Also given probabilistic nature of disks in seek and rotational delays
- Can characterize distribution of values of a random variable with discrete values using a **histogram**
 - Divides range between the min & max values into *buckets*
 - Histograms then plot the number in each bucket as columns
 - Works for discrete values e.g., number of I/O requests?
- What about if not discrete? Very fine buckets

4/12/2006

CS252 s06 Storage

56

Characterizing distribution of a random variable

- Need mean time and a measure of variance
- For mean, use **weighted arithmetic mean(WAM)**:
- f_i = frequency of task i
- T_i = time for tasks i

weighted arithmetic mean

$$= f_1 \times T_1 + f_2 \times T_2 + \dots + f_n \times T_n$$

- For variance, instead of standard deviation, use **Variance (square of standard deviation) for WAM**:
- **Variance** = $(f_1 \times T_1^2 + f_2 \times T_2^2 + \dots + f_n \times T_n^2) - WAM^2$
 - If time is milliseconds, Variance units are square milliseconds!
- Got a unitless measure of variance?

4/12/2006

CS252 s06 Storage

57

Squared Coefficient of Variance (C^2)

- $C^2 = \text{Variance} / WAM^2$
 - Unitless measure
- $C = \sqrt{\text{Variance}} / WAM = \text{StDev} / WAM$
- Trying to characterize random events, but to predict performance need distribution of random events where math is tractable
- Most popular such distribution is **exponential distribution**, where $C = 1$
- Note using constant to characterize variability about the mean
 - Invariance of C over time \Rightarrow history of events has no impact on probability of an event occurring now
 - Called **memoryless**, an important assumption to predict behavior
 - (Suppose not; then have to worry about the exact arrival times of requests relative to each other \Rightarrow make math considerably less tractable!)
- Most widely used exponential distribution is **Poisson**

4/12/2006

CS252 s06 Storage

58

Poisson Distribution

- Most widely used exponential distribution is **Poisson**
- Described by probability mass function:
Probability (k) = $e^{-a} \times a^k / k!$
 - where a = Rate of events \times Elapsed time
- If interarrival times are exponentially distributed and use arrival rate from above for rate of events, number of arrivals in time interval t is a **Poisson process**
- Time in Queue vs. Length of Queue?
- $\frac{1}{2} \times \text{Arithmetic mean} \times (1 + C^2)$

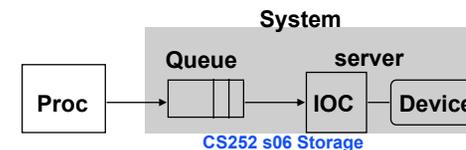
4/12/2006

CS252 s06 Storage

59

Summary

- **Disks: Aerial Density now 30%/yr vs. 100%/yr in 2000s**
- **TPC: price performance as normalizing configuration feature**
 - Auditing to ensure no foul play
 - Throughput with restricted response time is normal measure
- **Fault \Rightarrow Latent errors in system \Rightarrow Failure in service**
- Components often fail slowly
- Real systems: problems in maintenance, operation as well as hardware, software
- **Queuing models assume state of equilibrium: input rate = output rate**
- **Little's Law: $\text{Length}_{\text{system}} = \text{rate} \times \text{Time}_{\text{system}}$**
(Mean number customers = arrival rate \times mean service time)



4/12/2006

CS252 s06 Storage

60