CDA 5106 Advanced Computer Architecture 1

Module 8 | Storage Systems

Outline

- Magnetic Disks
- RAID
- Advanced Dependability/Reliability/Availability
- I/O Benchmarks, Performance and Dependability
- Intro to Queueing Theory
- Conclusion

Motivation: Who Cares About I/O?

- CPU Performance: 60% per year
- I/O system performance limited by *mechanical* delays (disk I/O)

  < 10% per year (IO per sec or MB per sec)
- Amdahl's Law: system speed-up limited by the slowest part!

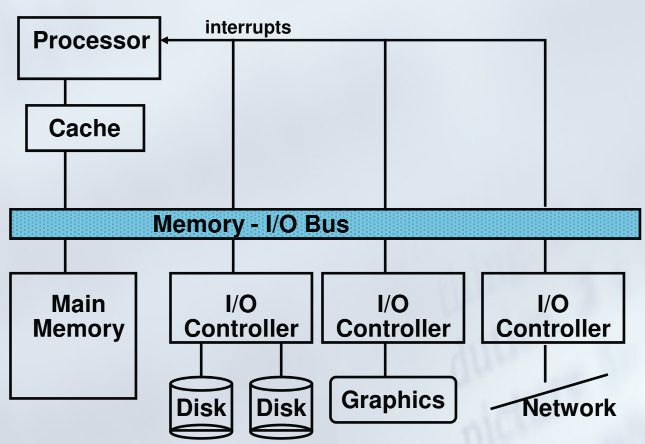
  10% IO & 10x CPU => 5x Performance (lose 50%)

  10% IO & 100x CPU => 10x Performance (lose 90%)
- I/O bottleneck:

  Diminishing fraction of time in CPU

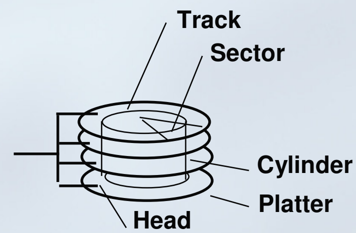
  Diminishing value of faster CPUs

I/O Systems



Magnetic Disks

- Purpose:
  - Long-term, nonvolatile storage
  - Large, inexpensive, slow level in the storage hierarchy
- Characteristics:
  - Seek Time (~8 ms avg)
- Transfer rate
  - About a sector per ms (5-15 MB/s)
  - Blocks
- Capacity
  - Gigabytes
  - Quadruples every 3 years (aerodynamics)

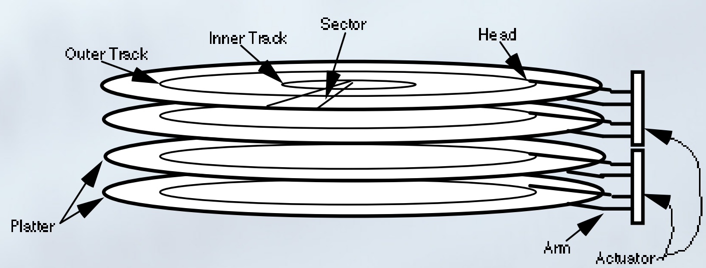


7200 RPM = 120 RPS => 8 ms per rev
avg rot. latency = 4 ms
128 sectors per track, 0.25 ms transfer per sector
1 KB per sector => 16 MB / s

Response time
= Queue + Controller + Seek + Rot + Xfer

Service time

Disk Device Terminology



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

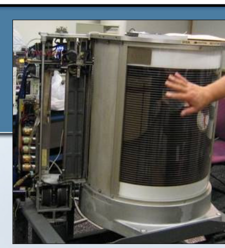
Order of magnitude times for 4K byte transfers:

Seek: 8 ms or less

Rotate: 4.2 ms @ 7200 rpm

Xfer: 1 ms @ 7200 rpm

Historical Perspective

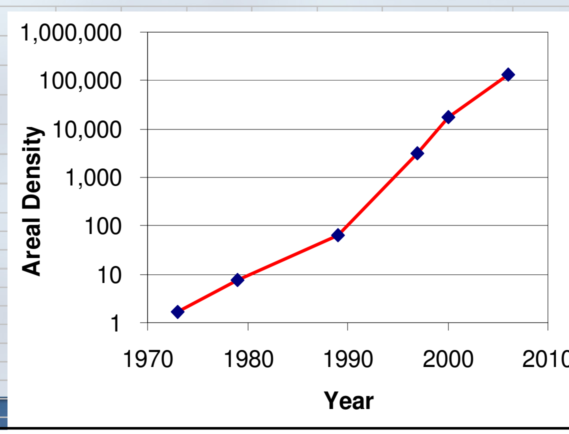


- 1956 IBM Ramac — early 1970s Winchester
  - For mainframe computers, proprietary interfaces
  - Steady shrink in form factor: 27 in. to 14 in.
- Form factor and capacity drives market more than performance
- 1970s developments
  - 5.25 inch floppy disk form factor (microcode into mainframe)
  - Emergence 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?

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

Year	Areal Density
1973	2
1979	8
1989	63
1997	3,090
2000	17,100
2006	130,000



I/O Benchmarks

- For better or worse, benchmarks shape a field
  - Processor benchmarks classically aimed at response time for a 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 make this work
- Classic metric is Transactions Per Second (TPS)

9

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 $\Rightarrow$ chaos
- 1984 Jim Gray (Tandem) distributed paper to Tandem + 19 other companies proposing 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

10

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

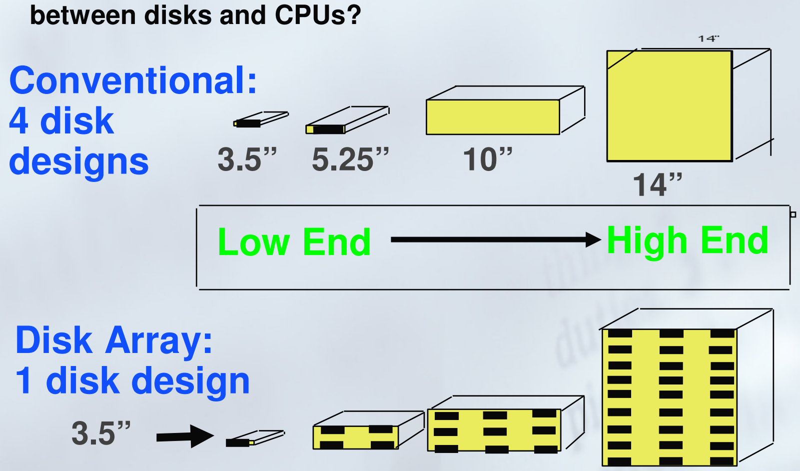
Year	Sequentially	Randomly (1 sector/seek)
1990	4 minutes	6 hours
2000	12 minutes	1 week(!)
2006	56 minutes	3 weeks (SCSI)
2006	171 minutes	7 weeks (SATA)

11

Use Arrays of Small Disks?

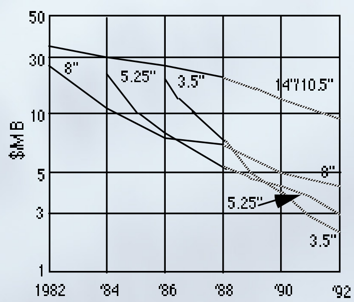
Katz and Patterson asked in 1987:

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



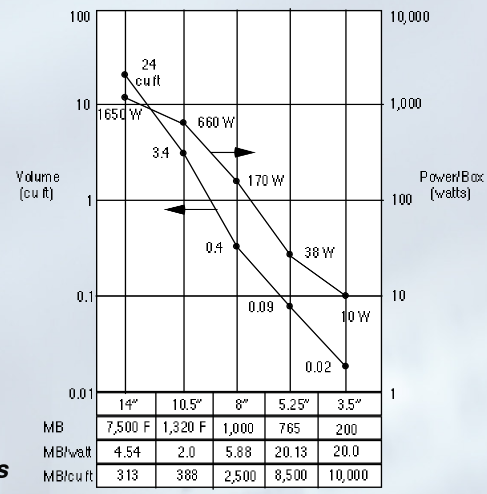
12

Advantages of Small Form Factor Disk Drives



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

Cost and Environmental Efficiencies



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	
MTTF	250 KHrs	50 KHrs	??? Hrs	6X
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?

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

15

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

16

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)

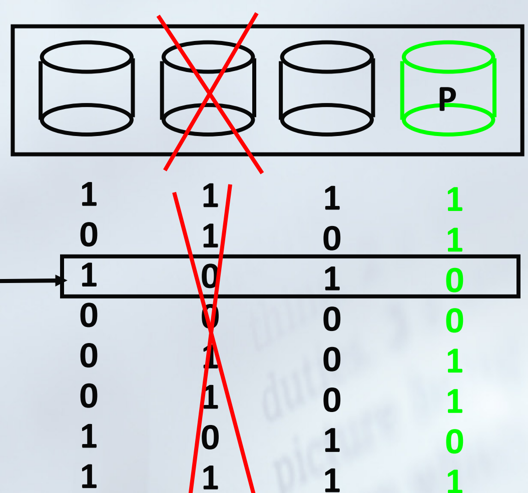
17

RAID 3: Parity Disk

10010011
11001101
10010011
. . .

logical record

Striped physical records



P contains sum of other disks per stripe mod 2 ("parity")
If disk fails, subtract P from sum of other disks to find missing information

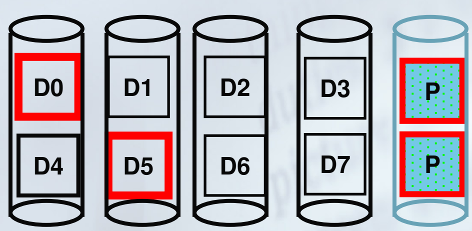
18

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

Inspiration for RAID 5

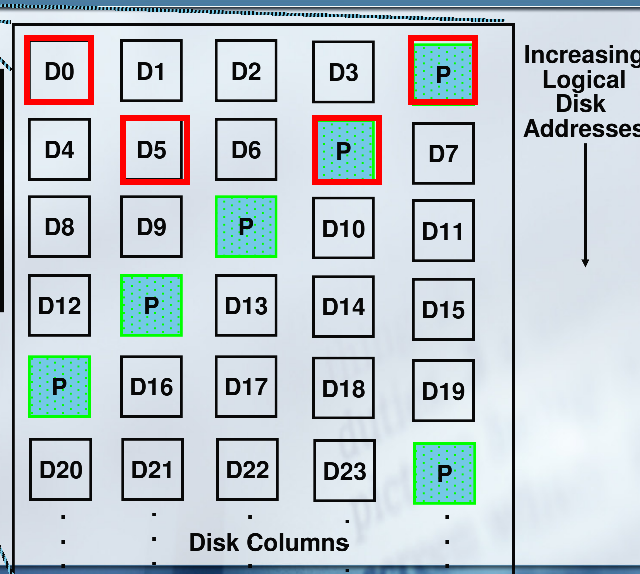
- RAID 4 works well for small reads
- Small writes (write to one disk):
  - Option 1: read other data disks, create new sum and write to Parity Disk
  - Option 2: since P has old sum, compare old data to new data, add the difference to P
- Small writes are limited by Parity Disk: Write to D0, D5 both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



RAID 5: High I/O Rate Interleaved Parity

Independent writes possible because of interleaved parity

Example: write to D0, D5 uses disks 0, 1, 3, 4

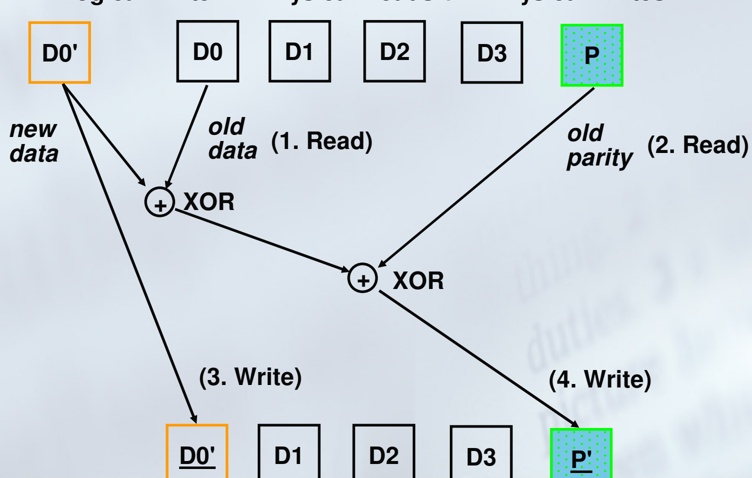


21

Problems of Disk Arrays: Small Writes

RAID-5: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes



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.

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

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

25

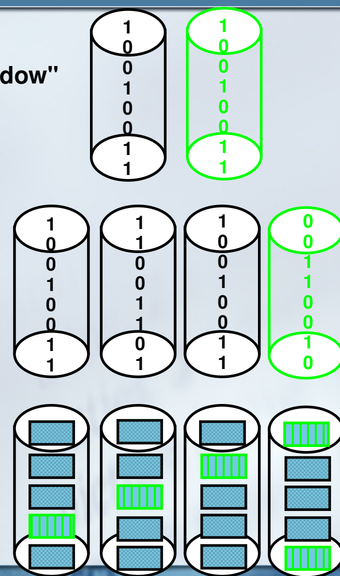
Berkeley History: RAID-I

- 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% non-PC disks sold in RAIDs

26

Summary: 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



27

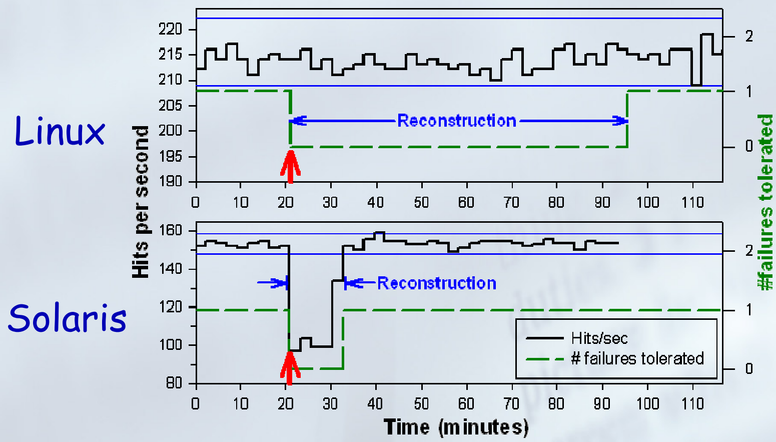
Reconstruction policy (2)

- 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)

28

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



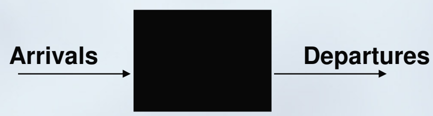
29

Review

- Disks: Arial 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 ⇒ Latent errors in system ⇒ Failure in service**
- Components often fail slowly
- Real systems: problems in maintenance, operation as well as hardware, software

30

Introduction to Queuing 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

31

Deriving Little's Law

- $\text{Time}_{\text{observe}}$ = elapsed time that we observe a system
- $\text{Number}_{\text{task}}$ = number of (overlapping) tasks 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

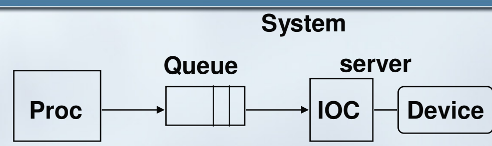
- $\text{Time}_{\text{accumulated}} / \text{Time}_{\text{observe}} = \text{Time}_{\text{accumulated}} / \text{Number}_{\text{task}} \times \text{Number}_{\text{task}} / \text{Time}_{\text{observe}}$

Then get Little's Law:

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

32

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

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 Time_{server}
- What is disk utilization if get 50 I/O requests per second for disk and average disk service time is 10 ms?
- Server utilization = 50/sec x 0.01 sec = 0.5
- Or server is busy on average 50% of time

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 to predict last part?
- To predict performance, need to know sometime about distribution of events

35

Distribution of Random Variables

- A variable is random if it takes one of a specified set of values with a specified probability
  - **Cannot know exactly next value, but may know 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

36

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
- Ti = time for tasks i

weighted arithmetic mean
  = $f1 \times T1 + f2 \times T2 + . . . + fn \times Tn$

- For variance, instead of standard deviation, use Variance (square of standard deviation) for WAM:
- Variance = $(f1 \times T1^2 + f2 \times T2^2 + . . . + fn \times Tn^2) - WAM^2$
  - If time is milliseconds, Variance units are square milliseconds
- Got a unitless measure of variance?

37

Squared Coefficient of Variance (C^2)

- C^2 = Variance / WAM^2
  $\Rightarrow$ C = sqrt(Variance)/WAM = StDev/WAM
  - Unitless measure
- Trying to characterize random events, but need distribution of random events with tractable math
- 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 not tractable!)

38

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 x Elapsed time
- If inter-arrival times exponentially distributed & use arrival rate from above for rate of events, number of arrivals in time interval *t* is a *Poisson process*

39

Time in Queue

- Time new task must wait for server to complete a task assuming server busy
  - Assuming it's a Poisson process
- Average residual service time = ½ x Arithmetic mean x (1 + C^2)
  - When distribution is not random & all values = average ⇒ standard deviation is 0 ⇒ C is 0
⇒ average residual service time = half average service time
  - When distribution is random & Poisson ⇒ C is 1 ⇒ average residual service time = weighted arithmetic mean

40

Time in Queue

- All tasks in queue ($Length_{queue}$) ahead of new task must be completed before task can be serviced
  - Each task takes on average $Time_{server}$
  - Task at server takes average residual service time to complete
- Chance server is busy is *server utilization*
  $\Rightarrow$ expected time for service is Server utilization $\times$ Average residual service time
- $Time_{queue}$ = $Length_{queue}$ x $Time_{server}$
  + Server utilization x Average residual service time
- Substituting definitions for $Length_{queue,}$ Average residual service time, & rearranging:

  $Time_{queue}$ = $Time_{server}$ x Server utilization/(1-Server utilization)

41

Time in Queue vs. Length of Queue

- $Length_{queue}$ = Arrival rate x $Time_{queue}$
  - Little's Law applied to the components of the black box since they must also be in equilibrium
- Given
  1. $Time_{queue}$ = $Time_{server}$ x Server utilization/(1-Server utilization)
  2. Arrival rate $\times$ $Time_{server}$ = Server utilization

$\Rightarrow$ $Length_{queue}$ = Server utilization^2/(1-Server utilization)

- Mean no. requests in queue? (If utilization is 50%)
- $Length_{queue}$ = $(0.5)^2$ / (1-0.5) = 0.25/0.5 = 0.5

$\Rightarrow$ 0.5 requests on average in queue

42

M/M/1 Queuing Model

- System is in equilibrium
- Times between 2 successive requests arriving, *"inter-arrival times"*, are exponentially distributed
- Number of sources of requests is unlimited "infinite population model"
- Server can start next job immediately
- Single queue, no limit to length of queue, and FIFO discipline, so all tasks in line must be completed
- There is one server
- Called M/M/1 (book also derives M/M/m)
  1. Exponentially random request arrival ($C^2 = 1$)
  2. Exponentially random service time ($C^2 = 1$)
  3. 1 server
  - *M* standing for Markov, mathematician who defined and analyzed the memoryless processes

43

Example

- 40 disk I/Os / sec, requests are exponentially distributed, and average service time is 20 ms

⇒ Arrival rate/sec = 40, $\text{Time}_{\text{server}}$ = 0.02 sec

1. On average, how utilized is the disk?

- Server utilization = Arrival rate × $\text{Time}_{\text{server}}$ = 40 x 0.02 = 0.8 = 80%

2. What is the average time spent in the queue?

- $\text{Time}_{\text{queue}} = \text{Time}_{\text{server}}$ x Server utilization/(1-Server utilization)
  = 20 ms x 0.8/(1-0.8) = 20 x 4 = 80 ms

3. What is the average response time for a disk request, including the queuing time and disk service time?

- $\text{Time}_{\text{system}} = \text{Time}_{\text{queue}} + \text{Time}_{\text{server}}$ = 80+20 ms = 100 ms

44

How much better with 2X faster disk?

- Average service time is 10 ms

  ⇒ Arrival rate/sec = 40, $Time_{server}$ = 0.01 sec

1. On average, how utilized is the disk?
   - Server utilization = Arrival rate × $Time_{server}$ = 40 x 0.01 = 0.4 = 40%
2. What is the average time spent in the queue?
   - $Time_{queue}$ = $Time_{server}$ x Server utilization/(1-Server utilization)

     = 10 ms x 0.4/(1-0.4) = 10 x 2/3 = 6.7 ms
3. What is the average response time for a disk request, including the queuing time and disk service time?
   - $Time_{system}$=$Time_{queue}$ + $Time_{server}$= 6.7 + 10 ms = 16.7 ms
   - 6X faster response time with 2X faster disk!

Ex: http://www.dcs.ed.ac.uk/home/jeh/Simjava/queueing/mm1_q/mm1_q.html

Value of Queuing Theory in practice

- Learn quickly do not try to utilize resource 100% but how far should back off?
- Allows designers to decide impact of faster hardware on utilization and hence on response time
- Works surprisingly well

Cross cutting Issues: Buses ⇒ point-to-point links and switches

Standard	width	length	Clock rate	MB/s	Max
(Parallel) ATA	8b	0.5 m	133 MHz	133	2
Serial ATA	2b	2 m	3 GHz	300	?
(Parallel) SCSI	16b	12 m	80 MHz (DDR)	320	15
Serial Attach SCSI	1b	10 m	--	375	16,256
PCI	32/64	0.5 m	33 / 66 MHz	533	?
PCI Express	2b	0.5 m	3 GHz	250	?

- No. bits and BW is per direction ⇒ 2X for both directions (not shown)
- Since use fewer wires, commonly increase BW via versions with 2X-12X the number of wires and BW

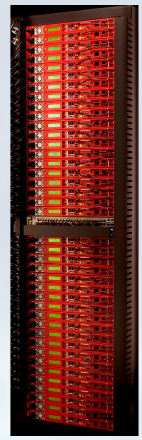
47

Storage Example: Internet Archive

- Goal of making a historical record of the Internet
  - Internet Archive began in 1996
  - Wayback Machine interface perform time travel to see what the website at a URL looked like in the past
  - Ex: http://web.archive.org/web/*/www.ucf.edu
- It contains over a petabyte (10^{15} bytes), and is growing by 20 terabytes (10^{12} bytes) of new data per month
- In addition to storing the historical record, the same hardware is used to crawl the Web every few months to get snapshots of the Interne.

48

Internet Archive Cluster



- 1U storage node PetaBox GB2000 from Capricorn Technologies
- Contains 4 500 GB Parallel ATA (PATA) disk drives, 512 MB of DDR266 DRAM, one 10/100/1000 Ethernet interface, and a 1 GHz C3 Processor from VIA (80x86).
- Node dissipates ≈ 80 watts
- 40 GB2000s in a standard VME rack,
  ⇒ 80 TB of raw storage capacity
- 40 nodes are connected with a 48-port 10/100 or 10/100/1000 Ethernet switch
- Rack dissipates about 3 KW
- 1 PetaByte = 12 racks

49

Estimated Cost

- Via processor, 512 MB of DDR266 DRAM, ATA disk controller, power supply, fans, and enclosure = $500
- 7200 RPM Parallel ATA drives holds 500 GB = $375
- 48-port 10/100/1000 Ethernet switch and all cables for a rack = $3000
- Cost $84,500 for a 80-TB rack
- 160 Disks are ≈ 60% of the cost

50

Estimated Performance

- 7200 RPM Parallel ATA drives holds 500 GB, has an average time seek of 8.5 ms, transfers at 50 MB/second from the disk. The PATA link speed is 133 MB/second
  - performance of the VIA processor is 1000 MIPS
  - operating system uses 50,000 CPU instructions for a disk I/O
  - network protocol stacks uses 100,000 CPU instructions to transmit a data block between the cluster and the external world
- ATA controller overhead is 0.1 ms to perform a disk I/O
- Average I/O size is 16 KB for accesses to the historical record via the Wayback interface, and 50 KB when collecting a new snapshot
- Disks are limit: ≈ 75 I/Os/s per disk, 300/s per node, 12000/s per rack, or about 200 to 600 Mbytes / sec Bandwidth per rack
- Switch needs to support 1.6 to 3.8 Gbits/second over 40 Gbit/sec links

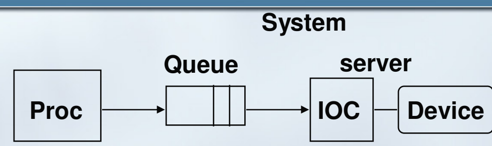
51

Estimated Reliability

- CPU/memory/enclosure MTTF is 1,000,000 hours (x 40)
- PATA Disk MTTF is 125,000 hours (x 160)
- PATA controller MTTF is 500,000 hours (x 40)
- Ethernet Switch MTTF is 500,000 hours (x 1)
- Power supply MTTF is 200,000 hours (x 40)
- Fan MTTF is 200,000 hours (x 40)
- PATA cable MTTF is 1,000,000 hours (x 40)
- MTTF for the system is 531 hours (≈ 3 weeks)
- 70% of time failures are disks
- 20% of time failures are fans or power supplies

52

Summary



- Little's Law: *$Length_{system}$ = rate x $Time_{system}$*
  (Mean number customers = arrival rate x mean service time)
- Appreciation for relationship of latency and utilization:
- $Time_{system}$= $Time_{server}$ +$Time_{queue}$
- $Time_{queue}$ = $Time_{server}$
  x Server utilization/(1-Server utilization)
- Clusters for storage as well as computation
- RAID: it's reliability, not performance, that matters for storage

Infosessions

- Anyone still hasn't done one?
  - Markov chains (intro)