

Building RAID Storage Systems

Lecture 9

NEU csg389, Spring 2005

Plan for today

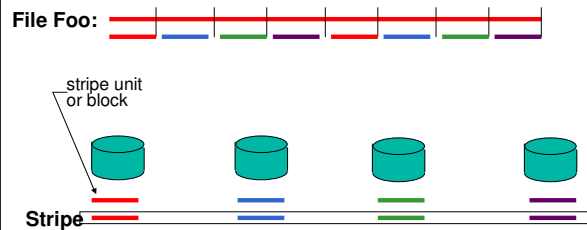
- Review
 - Ordering of metadata operations
 - Striping & redundancy
- Disk rebuild and reliability math
- Disk array controllers, parts and functions
 - increasing performance
 - increasing reliability
- Aggregation of storage
- Modeling & prediction

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

2

Disk Striping

- Interleave data across multiple disks
 - Large file streaming can enjoy parallel transfers
 - High throughput requests can enjoy load balancing



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

3

Disk Striping Details

- How disk striping works
 - Break up total space into fixed-size stripe units
 - Distribute the stripe units among disks in round-robin
 - Compute location of block #B as follows
 - $\text{disk\#} = B \% N$ ($\% = \text{modulo}$, $N = \# \text{ of disks}$)
 - $\text{LBN\#} = B / N$ (computes the LBN on given disk)
- Key design decision: picking the stripe unit size
 - too big: no parallel transfers and imperfect load balancing
 - too small: small transfers span stripe unit boundaries
 - also, should be a multiple of block size to assist alignment

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

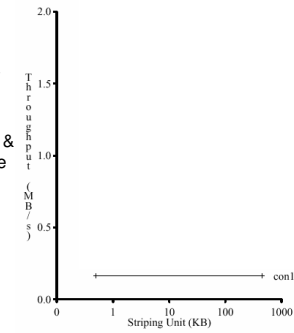
4

Striping Issues

- Striping distributes load across all the disks
 - Significantly improves load balancing
- However, watch out for multiple disk accesses
 - Small access > 1 strip unit require multiple disk accesses
 - Small accesses < 1 strip unit that cross strip unit boundaries
 - Performance is bounded by “slower” access
- What’s the “optimal” strip unit?

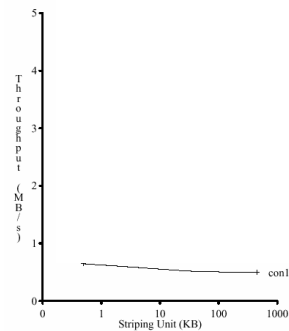
Picking a Striping Unit

- Exponential distribution with mean of 4KB
 - Concurrency = 1 means the system is idle when a request is launched
- 16 drives is simulations
- For a small request size (4K) & concurrency = 1, the strip size doesn't matter
 - Why?



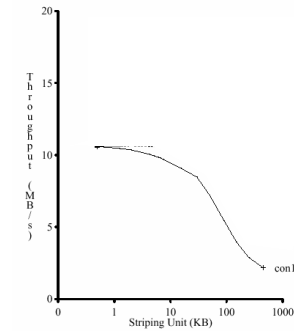
Picking a Striping Unit (2)

- Exponential distribution with mean of 16KB
- For concurrency = 1, the strip size matters only a little
 - Why?



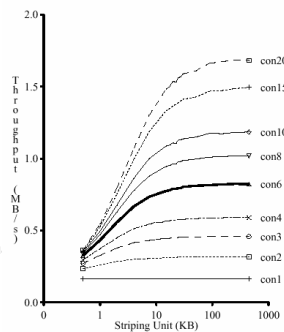
Picking a Striping Unit (2b)

- Exponential distribution with mean of 400KB
- For concurrency = 1, stripe size matters a lot
 - Why?



Picking a Striping Unit (3)

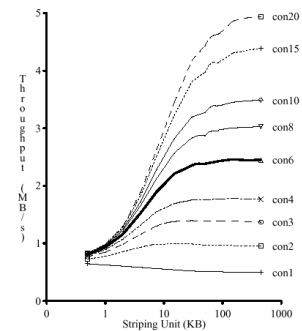
- Exponential distribution with mean of 4KB
 - With concurrency from 1--20 outstanding requests
 - Concurrency = 1 means the system is idle when a request is launched
- For a small request size (4K) & concurrency = 1, the strip size doesn't matter
 - Why?
- For larger concurrency, perform increases with larger strip unit
 - Why?



(a) size *exp4k*

Picking a Striping Unit (4)

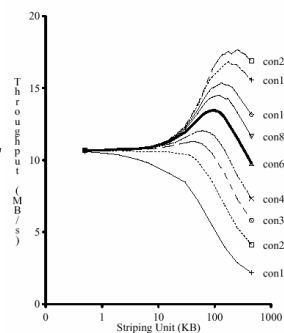
- Exponential distribution with mean of 16KB



(b) size *exp16k*

Picking a Strip Unit (5)

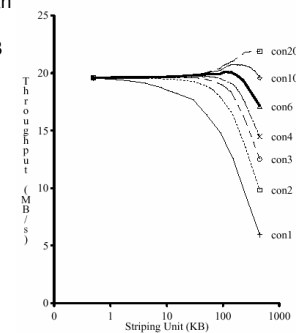
- Normal distribution with mean request size = 400KB and standard deviation of 400KB
- For each level of concurrency, there is optimal striping unit
 - For concurrency=1, the optimal striping unit is 0.5K
 - For concurrency=20, the optimal striping unit is ~400K



(c) size *norm400k*

Picking a Strip Unit (6)

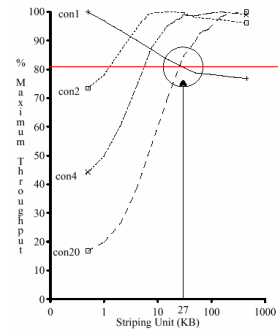
- Normal distribution with mean request size = 1.5 MB and standard deviation of 1.5 MB
- Again, every level of concurrency has an optimal striping unit



(d) size *norm1.5m*

Picking a Striping Unit (7)

- Both concurrency and request size are important facts when picking the optimal striping unit
- In these 3 graphs (see next slide), there is an optimal point for a given request size
- Could also compute optimal point for a given level of concurrency

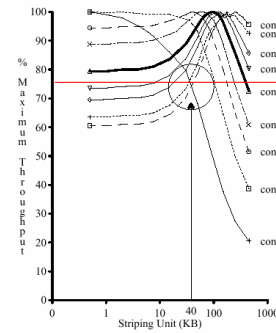


(b) size *exp16k*

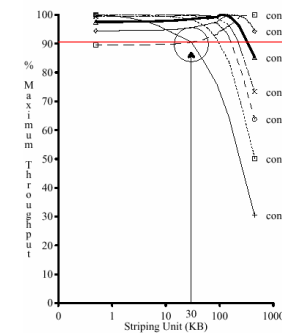
NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

13

Picking a Striping Unit (8)



(c) size *norm400k*

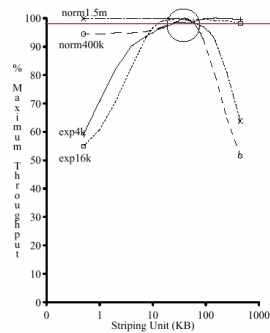


(d) size *norm1.5m*

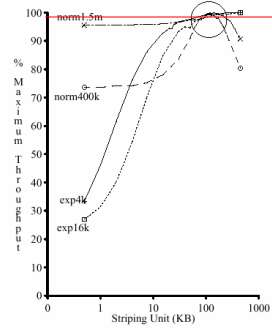
NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

14

Picking a Striping Unit (9)



(b) concurrency 3



(c) concurrency 8

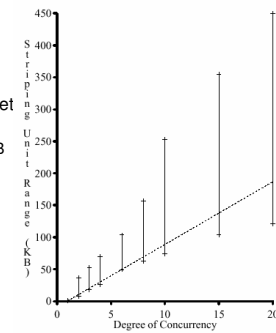
NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

15

Picking a Striping Unit (10)

- Using the data from the previous graphs
 - This graph plots the range of striping unit sizes which have 95%+ throughput
 - Fitting a line to this graph, we get

$$SU = 9.8KB * (Concurrency-1) + 0.5KB$$



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

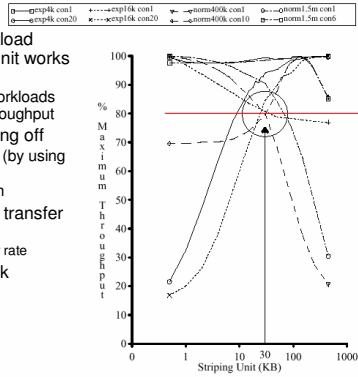
16

Picking a Striping Unit (11)

- For a wider range of workload options, a 30KB striping unit works best
 - At this point, all of the workloads achieve at least 80% throughput
- Picking a strip unit is trading off
 - Decreased transfer time (by using more disks per request)
 - Increasing disk utilization
- The benefit of decreasing transfer time is apx

$$\frac{\text{striping unit} / \text{disk transfer rate}}{\text{Average positioning time}}$$
- The increasing cost of disk positioning is apx

$$\text{Average positioning time}$$

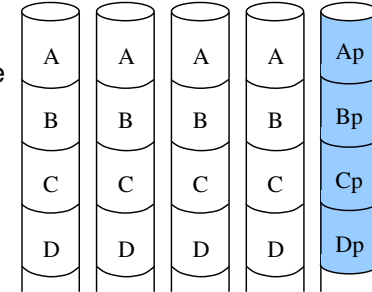


NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

17

Simplest approach: Parity Disk

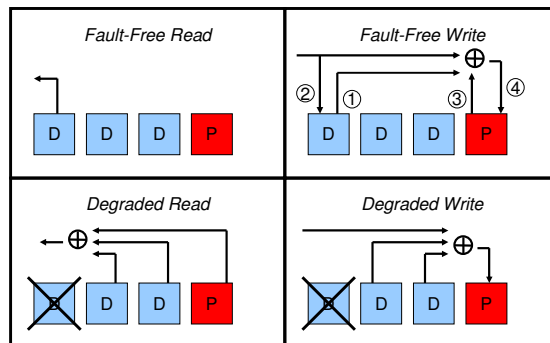
- One extra disk
- All writes update parity disk
 - potential bottleneck



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

18

Updating and Using the Parity



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

19

The Disk Array Matrix

	Independent	Fine Striping	Coarse Striping
None	JBOD		RAID0
Replication	Mirroring RAID1		RAID0+1
Parity Disk		RAID3	RAID4
Striped Parity			RAID5

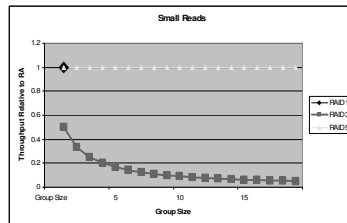
NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

20

RAID Performance (relative to RAID 0)

Level	Small Read	Small Write	Large Read	Large Write	Efficiency
0	1				
1	1				
3	1/G				
5	1				

G is the # of drives in the error-correction group



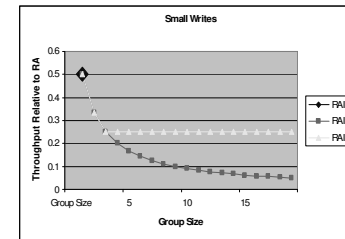
NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

21

RAID Performance (relative to RAID 0)

Level	Small Read	Small Write	Large Read	Large Write	Efficiency
0	1	1			
1	1	1/2			
3	1/G	1/G			
5	1	max(1/G, 1/4)			

G is the # of drives in the error-correction group



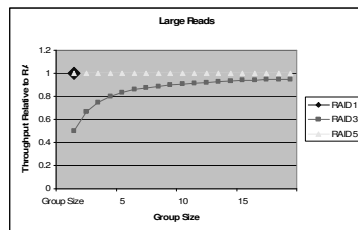
NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

22

RAID Performance (relative to RAID 0)

Level	Small Read	Small Write	Large Read	Large Write	Efficiency
0	1	1	1		
1	1	1/2	1		
3	1/G	1/G	(G-1)/G		
5	1	max(1/G, 1/4)	1		

G is the # of drives in the error-correction group



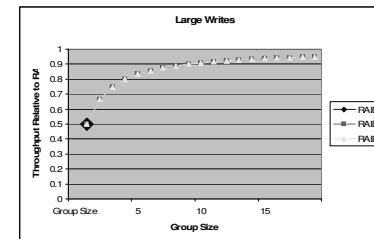
NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

23

RAID Performance (relative to RAID 0)

Level	Small Read	Small Write	Large Read	Large Write	Efficiency
0	1	1	1	1	1
1	1	1/2	1	1/2	1/2
3	1/G	1/G	(G-1)/G	(G-1)/G	(G-1)/G
5	1	max(1/G, 1/4)	1	(G-1)/G	(G-1)/G

G is the # of drives in the error-correction group



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

24

Reliability Without Rebuild

- 200 data drives with $MTBF_{drive}$
 - $MTTDL_{array} = MTBF_{drive} / 200$
- Add 200 drives and do mirroring
 - $MTBF_{pair} = (MTBF_{drive} / 2) + MTBF_{drive} = 1.5 * MTBF_{drive}$
 - $MTTDL_{array} = MTBF_{pair} / 200 = MTBF_{drive} / 133$
- Add 50 drives, each with parity across 4 data disks
 - $MTBF_{set} = (MTBF_{drive} / 5) + (MTBF_{drive} / 4) = 0.45 * MTBF_{drive}$
 - $MTTDL_{array} = MTBF_{set} / 50 = MTBF_{drive} / 111$

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

25

Rebuild: Restoring Redundancy After Failure

- After a drive failure
 - data is still available for access
 - but, a second failure is BAD
- So, should reconstruct the data onto a new drive
 - on-line spares are features of high-end disk arrays
 - reduce time to start rebuild
 - must balance rebuild rate with foreground performance impact
 - a performance vs. reliability trade-offs
- How data is reconstructed
 - Mirroring: just read good copy
 - Parity: read all remaining drives (including parity) and compute

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

26

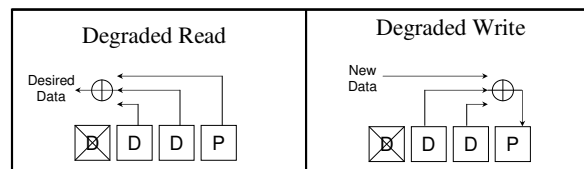
Reliability Consequences of Adding Rebuild

- No data loss, if fast enough
 - That is, if first failure fixed before second one happens
- New math is...
 - $MTTDL_{array} = MTBF_{firstdrive} * (1 / \text{prob of } 2^{nd} \text{ failure before repair})$
 - ... which is $MTTR_{drive} / MTBF_{seconddrive}$
- For mirroring
 - $MTBF_{pair} = (MTBF_{drive} / 2) * (MTBF_{drive} / MTTR_{drive})$
- For 5-disk parity-protected arrays
 - $MTBF_{set} = (MTBF_{drive} / 5) * (MTBF_{drive} / 4 / MTTR_{drive})$

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

27

Updating and Using the Parity



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

28

Three Modes of Disk Array Operation

- Normal mode
 - everything working; maximum efficiency
- Degraded mode
 - some disk unavailable
 - must use degraded mode operations
- Rebuild mode
 - reconstructing lost disk's contents onto spare
 - degraded mode operations plus competition with rebuild

Mechanics of Rebuild

- Background process
 - use degraded mode read to reconstruct data
 - then, write it to replacement disk
- Implementation issues
 - interference with foreground activity
 - rebuild is important for reliability
 - foreground activity is important for performance
 - using the rebuilt disk
 - for rebuilt part, reads can use replacement disk
 - must balance performance benefit with rebuild interference

Refresh: The Disk Array Matrix

	Independent	Fine Striping	Coarse Striping
None	JBOD		RAID0
Replication	Mirroring RAID1		RAID0+1
Parity Disk		RAID3	RAID4
Striped Parity			RAID5

Disk Array Subsystems

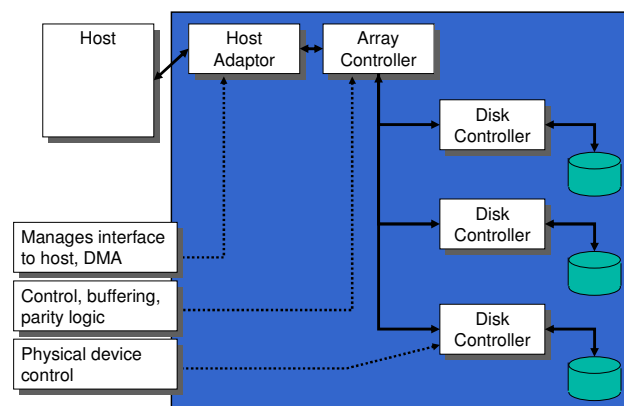
Disk Array Subsystems

- Sets of disks managed by a central authority
 - e.g., file system (within OS) or disk array controller
- Data distribution
 - squeezing maximum performance from the set of disks
 - several simultaneous considerations
 - intra-access parallelism: parallel transfer for large requests
 - inter-access parallelism: concurrent accesses for small requests
 - load balancing for heavy workloads
- Redundancy scheme
 - achieving fault tolerance from the set of disks
 - several simultaneous considerations
 - space efficiency
 - number/type of faults tolerated
 - performance

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

33

Disk Array Controller components



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

34

Improving Reliability

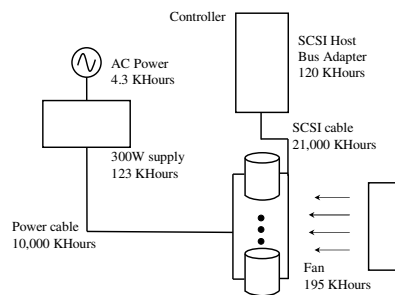
- Addressing potential failure of other components
- Array component organization
 - put redundancy orthogonal to correlated disk hook-ups
- In-progress writes
 - NVRAM
 - write-ahead logging
- End-to-end integrity checks
 - error detection codes added by controller

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

35

Arrays Contain Support Hardware

- must protect external power quality first
- combined effects of other components rival disk failures



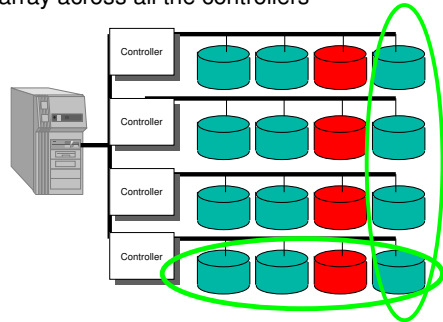
Schulze, Compton, 1989

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

36

Redundancy Orthogonal to Shared Paths

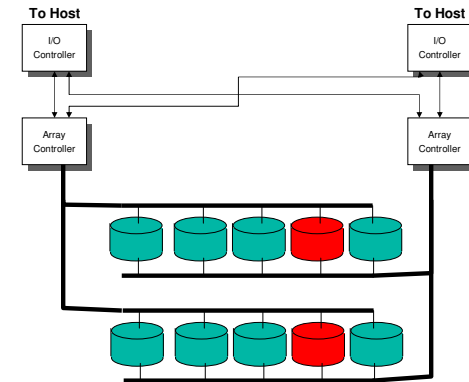
- Two 4-disk array setups
 - A) one array per controller
 - B) each array across all the controllers



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

37

Dual-Path Everything

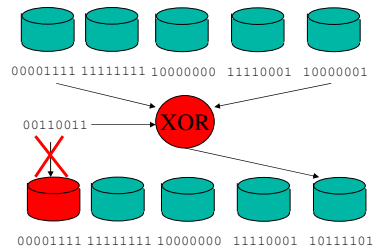


NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

38

Disk Array Integrity

- Disk arrays require writes to multiple disks
 - What happens if the array controller crashes during a write
 - Example, during the update operation, the parity disk is written, but the new data is not
 - When the controller comes back up the data will be inconsistent



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

39

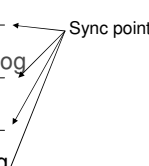
Disk array integrity (cont.)

- Why would a RAID system crash
 - Power failure
 - Earthquake
- Solution:
 - Log all writes before committing them to disks
 - Log all writes to NVRAM before writing to disk
 - After the write to disk(s) completes, update the log
 - If there is a failure, the system uses the log to return the array to a consistent state

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

40

Disk array integrity (cont)

- 2-phase commit process
 - Read drives to obtain old data and parity
 - Compute new parity
 - Write new block and parity to log
 - Write data and parity to disks
 - Mark operation complete in log
- 

Many other integrity mechanisms

- EDC/ECC on memories and buses
 - to protect against bit flips and such
- “Scrubbing” to catch grown defects
 - before they found during a rebuild
- Checksum labels on each block
 - to proactively detect problems caused by disk bugs

Improving Performance

- Hardware assists for XOR
 - controller ASIC support
 - common approach: a combining memory
 - drive-based XOR operations
 - Data drive
 - $\text{Mask} = \text{oldData} \text{ XOR } \text{newData}$
 - Parity Drive
 - $\text{Write}(\text{Mask XOR oldParity})$
 - Avoids having to ship oldParity to server or HBA
 - Reduces Bandwidth
- Cache memory

Buffering/Caching in Array Controllers

- Writes
 - signal completion once data is in array buffers
 - must deal with reliability issue (RAM failure, power loss)
 - benefits
 - all writes finish at RAM speed instead of disk speed
 - many writes may be coalesced and/or concatenated
 - parity updates can be combined
 - propagate to disks in background
 - redundancy info updates (even Read-Modify-Write)
- Reads
 - basic read caching and prefetching
 - advanced techniques
 - caching of parity blocks can eliminate need to read old parity on writes

Synchronized Spindles

- Disk Synchronization
 - Sometimes the disks work together
 - Full stripe read/write
 - Sometimes the disks work independently
 - Small read/write
 - If an entire stripe is read, then all of the disk heads will be on the same track
 - All of the disks will have the SAME seek time for the next full-stripe request
 - However, each disk could be spinning at a slightly different rate
 - Therefore, each disk may have a different rotational latency
 - Does this matter?
 - What's the Average Rotational Latency for the entire set of drives?
 - $Avg\ Rot\ Lat_{set} = N/(N+1) * Rotational\ Latency$
 - For a large number of drives, $Avg\ Rot\ Lat_{set} \sim Rotational\ Latency$

Synchronized Spindles (cont)

- Disk Synchronization (cont)
 - If an access is not across an entire stripe, then each disk head will probably be in a different part of the disk
 - Therefore, need to synchronize both seek and rotation
 - By simulation, Chen found that not having this synchronization added apx 3.5 ms (~12%) to an average seek + average rotation
 - What about the 2nd disk access
 - Remember, RAID 5 does a read-modify-write to both the data and parity drives
 - Since the first access has already moved the head to the correct track, the drives end up on the same track
 - But each must wait an entire rotation to perform the write

EMC Symmetrix 8000 (in year 2000)

- Symmetrix 8000
 - Holds up to 384 disks (73 GB/disk – 28 Terabytes)
 - RAID 1 or a RAID5 variant called RAID-S
 - Group size of 4 or 8 drives
- Architecture
 - 4 internal busses (480 MB/sec each with ECC)
 - 4 cache boards (1-8GB each)
 - 16 Disk Directors
 - Connect internal busses to disks
 - Two 333 MHz PowerPC750 processors with 16 MB of RAM
 - Each uses (redundant) SCSI controller
- Special disks
 - 512 byte data + extra bytes for ECC
 - Multiple priorities for reordering of requests in the disk queue

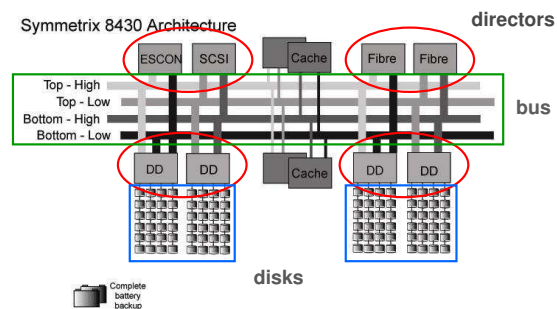
EMC Symmetrix 8000 (2)

- Performance Optimizations
 - For mirroring, Symmetrix watches disk access patterns to decide how to issue data requests
 - Interleaved – mirrors alternate which cylinders they serve
 - Helps with _____ accesses
 - How do you express cylinders?
 - Mixes – one disk serves the first half of the data, the other disk serves the second half
 - Helps with _____ I/O
 - What's the effective stripe unit size?
- Migration of partitions to balance load

EMC Symmetrix 8000 (3)

- Fault Tolerance
 - Cache blocks (32KB) plus ECC
 - ECC on all transfers between cache/host and disk/cache
- Memory Scrubbing
 - All cache locations are periodically read and rewritten to correct for single bit errors (and to detect double bit errors)
 - bad regions of memory with unrecoverable errors are fenced off
- Disk scrubbing
 - All disk locations are periodically read and rewritten to correct for errors
 - Bad sectors or tracks are fenced off
- XOR computation is performed inside the drives

Disk array physical components

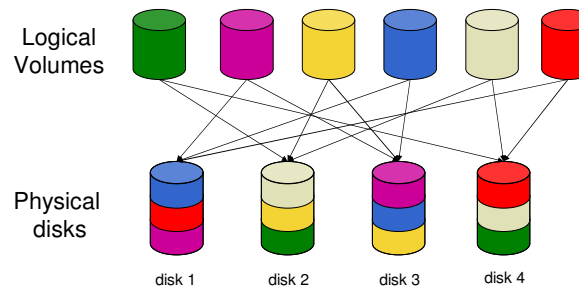


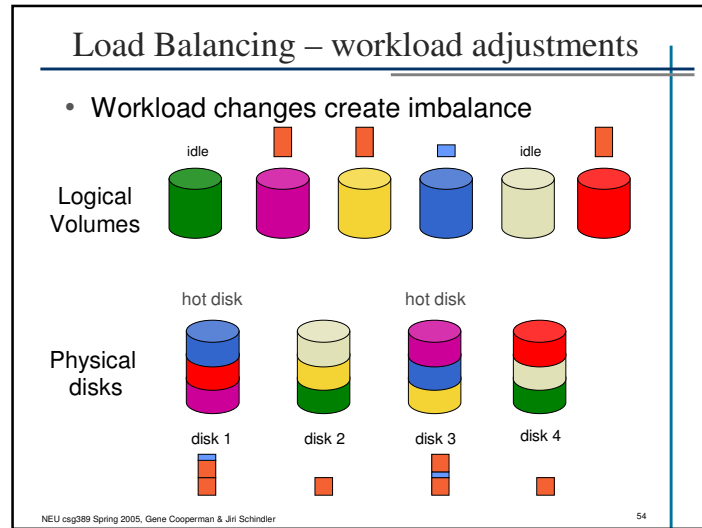
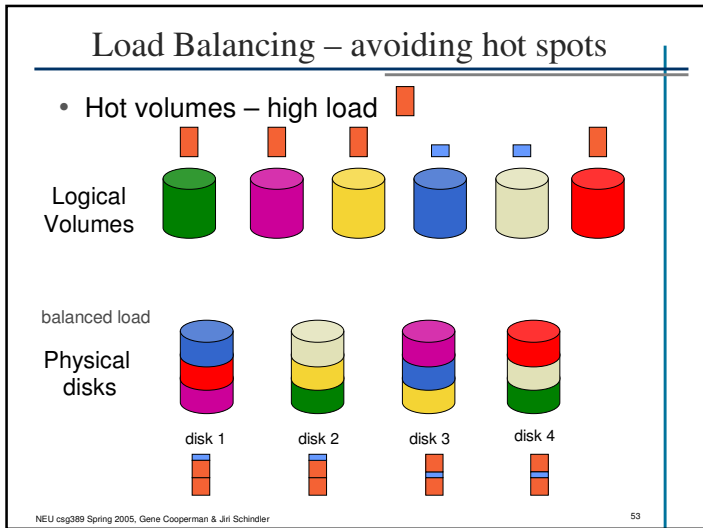
Aggregation into Logical Volumes

- Logical volumes
 - A single device presented to the client
 - A LV is mapped to a single Front-end director
 - created from several hyper volumes (parts of disks)
 - RAID-1
 - RAID-S (variant of RAID-5)
- Physical disks
 - sliced into "partitions" (called hyper volume)
 - one disk holds data from many volumes

Mapping Logical Volumes to disks

- Spread LV across multiple disks



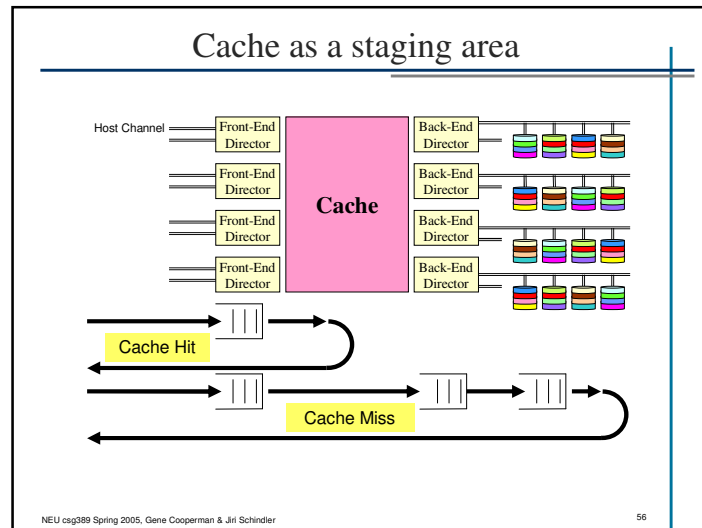


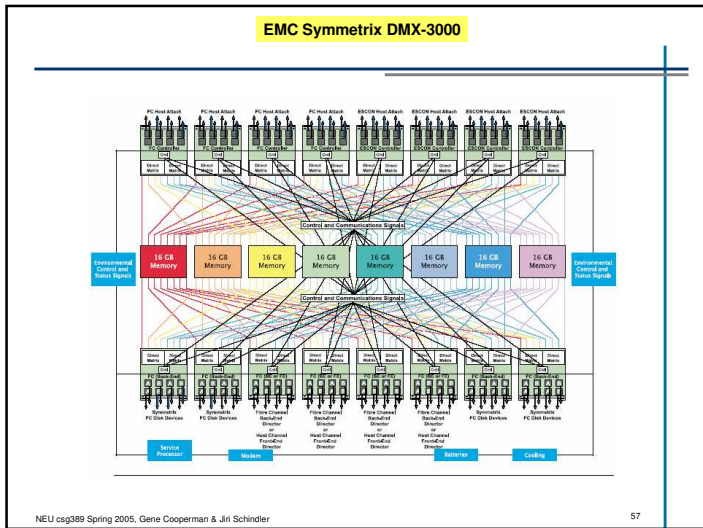
Caching

Shared cache among LVs


- individual pages (32KB)
- LRU policy
- Goal: let all I/Os be serviced from the cache
 - separate back-end traffic from front-end
 - discover sequential accesses
 - engage prefetching
 - lazy write-back
- Challenges
 - NVRAM backed by batteries
 - need enough power to keep data
 - ... and write them all to the disk...
 - 384 disks – a lot of power!

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler 55





Some numbers...



DMX3000

Drives	Capacity (raw)	Drive Channels	Global Memory Directors	Maximum Global Memory	Connectivity*
288	42 TB	32/64 x 2 Gb Fibre Channel	4-8	256 GB	96 x 2 Gb FC 96 x ESCON 48 x 2 Gb FICON 8 x GigE SRDF 48 x iSCSI

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler 58

Predicting Performance With Models

NEU csg389, Spring 2005

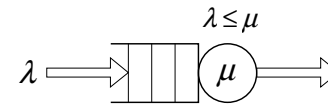
- ### More Performance Required
- Add another disk drive
 - Distribute (spread) load among two disks
 - mirroring
 - striping
 - Replace the current disk, with a faster one
 - Handle more I/Os
- But what if you need another faster component?
- Bottleneck analysis (homework)
 - More rigorous treatment today
- NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler 60

Need to Know

- Workload
 - measure or predict by calculation
- System model
 - interactions among components
- Individual device characteristics

Queueing Theory 0.101

- Disclaimer: No proofs in this lecture
- A basic component
 - service center
 - completion/service rate of μ jobs per unit time
 - queue with jobs waiting to be processed
 - arrival rate of λ jobs per unit time
 - Large-enough size, but



Some More Definitions

- Mean service time: $E[S] = \frac{1}{\mu}$
- Metrics
 - Response time: $E[T_s] = E[T_q] + E[S]$
 - Device throughput: $X = \frac{C}{T}$ # of completions
observation period
 - Device utilization: $U = \frac{B}{T}$ busy time
observation period

Utilization vs. Throughput

- Express throughput in terms of utilization

$$X = \frac{C}{T} = \frac{C}{B} \frac{B}{T} = \frac{C}{B} U \quad \frac{B}{C} = E[S] = \frac{1}{\mu}$$

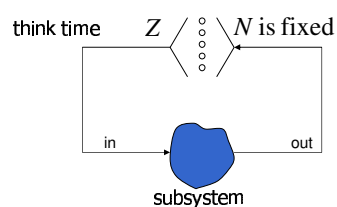
$$X = \mu \cdot U$$

- Utilization law: $U = X \cdot E[S]$

- Intuitively:

$$U = \frac{1/\mu}{1/\lambda} \quad \begin{array}{l} \text{avg. service time} \\ \text{avg. time between arrivals} \end{array} \quad U = \frac{\lambda}{\mu}$$

Closed System



number of jobs
in the system

X depends on μ

$\uparrow X \Leftrightarrow \downarrow E[T_s]$

- Little's law for closed systems: $N = X \cdot E[T_s]$

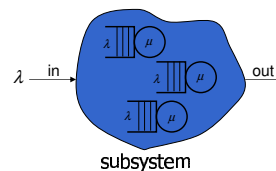
NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

65

Open System

- Little's law for open systems

$$E[N] = \lambda \cdot E[T_s]$$



- Arrival rate λ is important
- Various distributions of μ and λ
 - Usually exponential Poisson arrival process
- X and response time $E[T_s]$ unrelated

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

66

Open vs. Closed System

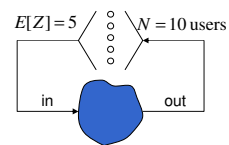
- We limit our analysis to closed systems
 - Easier computations
 - No need to know about distributions of λ and μ
 - Operational analysis works only for closed systems
 - Must know the "other side" of the system
 - Think times, batched systems etc.
- Analysis in practice is all about open systems
 - Observations made at the system
 - More powerful analysis
- Open research questions: open vs. closed

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

67

Example 1

- Given a system with number of users $N=10$, think time $E[Z]=5$ seconds, expected response time $E[R]=15$ seconds. Calculate the throughput X .



$$N = X \cdot E[T_s]$$

$$N = X (E[Z] + E[R])$$

$$X = \frac{N}{E[Z] + E[R]}$$

$$X = \frac{10}{5 + 15} = 0.5 \text{ interactions/second}$$

- Response time law: $E[R] = \frac{N}{X} - E[Z]$

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

68

Operational Laws

- Forced flow law: $X_i = X \cdot E[V_i]$
- Device demand: $C_i = C \cdot E[V_i]$
 $E[V_i]$ Expected number of visits to device i
- Bottleneck law: $E[D_i] = E[V_i] \cdot E[S_i]$
 $E[D_i] = \frac{B_i}{C}$
 $U_i = X \cdot E[D_i]$

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

69

Asymptotic Bounds

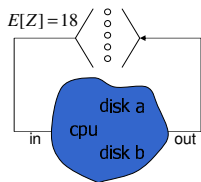
- Demands: $D = \sum_{i=1}^m E[D_i]$
 $D_{max} = \max(E[D_i])$
- Bounds for throughput: $X \leq \min\left(\frac{N}{D + E[Z]}, \frac{1}{D_{max}}\right)$
- Bounds for resp.time: $E[R] \geq N \cdot D_{max} - E[Z]$

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

70

Example 2

- Given a system with think time $E[Z]=18$ s, $E[D_{cpu}]=5$ s, $E[disk_a]=4$ s, $E[disk_b]=3$ s. Find the throughput X and $E[R]$ as a function of N .



$$D = 5 + 4 + 3 = 12$$

$$D_{max} = 5 \quad \text{bottleneck device is CPU}$$

$$X \leq \min\left(\frac{N}{30}, \frac{1}{5}\right)$$

$$E[R] \geq \max(12, 5N - 18)$$

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

71

Modeling - Lessons Learned

- Simple observations give powerful analysis
 - Prediction about performance
 - Number of users supported
 - Demand and throughput of individual components
 - No need to know about distributions of μ and λ
- Workload in open and closed systems
 - Closed – interactive and batch
 - Open – when whole system cannot be observed
- Next: building an open-system model

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

72

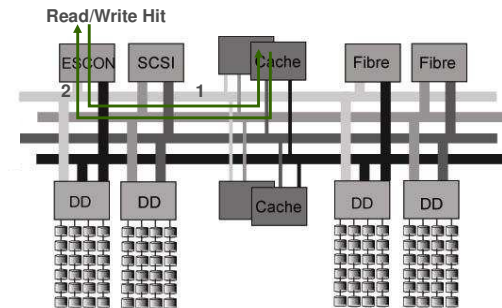
Workload Characterization

- Deciding on open vs. closed is not enough, we also need to characterize the workload bef
 - Recall from stripe-unit lecture how we characterized workload
- Express job size in terms of units (e.g., KB)
 - $\mu=4$ jobs/s of size 1KB or $\mu=2$ jobs/s of size 2KB
- Open systems
 - $\lambda_{\text{size } 1}=1$ jobs/s, $\lambda_{\text{size } 2}=0.5$ jobs/s
 - Arrival and completion rate distributions
 - Many theorems (and proofs) for exponential dist.
 - But the reality may be different
 - heavy-tail distribution for web workloads [Harchol-Balter99]
 - A lot of research

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

73

Cache Hit (Reads and Writes)

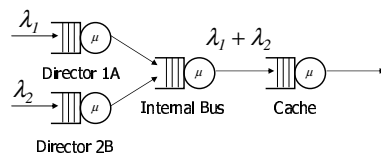


NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

74

Model for Cache Hits

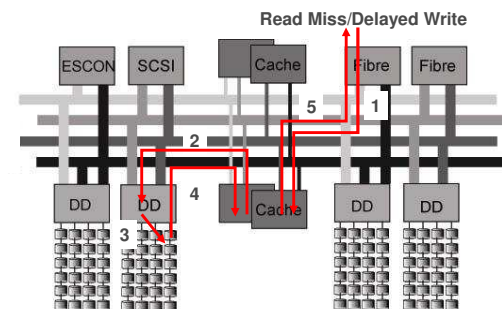
- Workload analyzer
 - 2 front-end directors, bus, cache
 - job sizes (just one in this example)
 - arrival rates for each job of one size
 - Can involve several LVs



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

75

Cache Miss

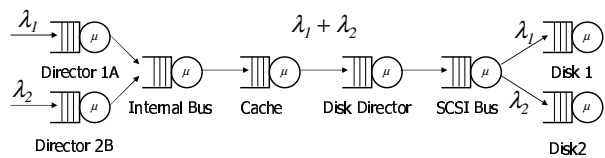


NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

76

Model for Cache Misses

- Workload analyzer
 - 2 front-end directors, bus (interconnect), cache, two disks
 - one job size
 - arrival rates for each job size to each LV



NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

77

Simulation vs. Analytical Model

- Analytical Models
 - Results valid only if the assumptions are
 - Arrival and completion rate distributions
 - "Results within 15%-20% of the reality are good" [Lazowska]
 - How to model a disk analytically?
- Simulators
 - Take a trace and run it through the simulator
 - Can give much more precise results
 - Time consuming (everything in software)
 - 200-5000 instructions/s (real processor vs. 1000s of MIPS)

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

78

Next week

- Performance analysis of disk array systems
 - guest lecture by Peter Lauterbach
- March 24th
 - visit to EMC
 - Meet at 9am, van leaves at 9:10 am
 - talk by Bob Salomon
 - tour of Interoperability lab
 - visit to Franklin assembly facility
 - Will be back by 3pm

NEU csg389 Spring 2005, Gene Cooperman & Jiri Schindler

79