

6.4 Multiple Disks

RAID

- what can be done if you lose an entire disk?



Copyright © William C. Cheng

Operating Systems - CSCI 402

Benefits of Multiple Disks

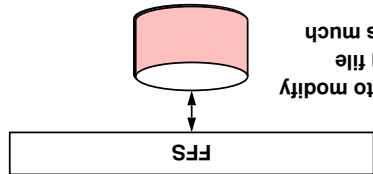
- They hold more data than one disk does
- Data can be stored *redundantly* so that if one disk fails, they can be found on another
- increase reliability
- increase availability
- Data can be spread across multiple drives, allowing *parallel* access
- increase effective access time



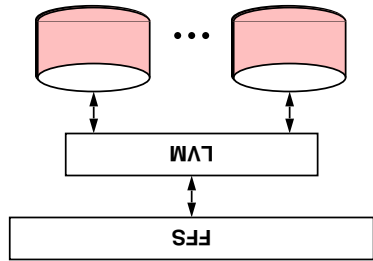
Copyright © William C. Cheng

Operating Systems - CSCI 402

Logical Volume Manager



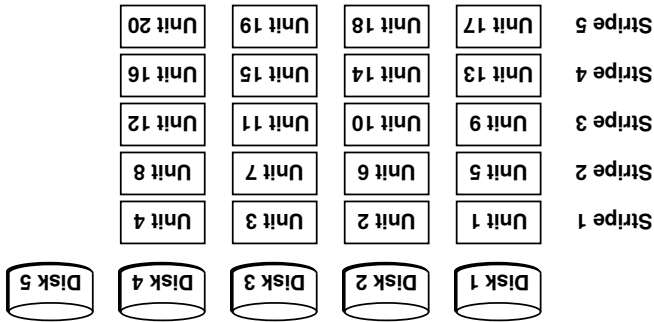
Try not to modify existing file systems much



Copyright © William C. Cheng

Operating Systems - CSCI 402

Striping



Ex: *stripe width* = 4

- theoretically, a *striping unit* can be a bit (i.e., *bit-interleaving*)
- pack these bits into disk blocks and store on disk



Copyright © William C. Cheng

Operating Systems - CSCI 402

Parallel Disks

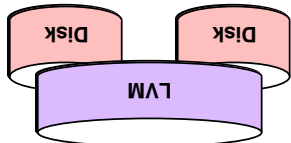
- Advantages
 - increase parallelism
 - can retrieve blocks belonging to multiple files simultaneously if they are on different disks
- Disadvantages
 - reduced access time if a block is spread over multiple disks
 - seek in parallel, same rotational latency on all disks



Copyright © William C. Cheng

Operating Systems - CSCI 402

Logical Volume Manager



- Spanning
 - two real disks appear to file system as one large disk
- Mirroring
 - file system writes redundantly to both disks
 - reads from one



Copyright © William C. Cheng

Operating Systems - CSCI 402

Copyright © William C. Cheng

RAID to the Rescue

- RAID: Redundant Array of Inexpensive Disks (as opposed to Single Large Expensive Disk: SLED)
- combine *striping* with *mirroring*
- 5 different variations originally defined
- RAID level 1 through RAID level 4 developed by IBM
- RAID level 5 developed by UC Berkeley
- RAID level 0: pure striping (numbering extended later)
- RAID level 1: pure mirroring

11

Copyright © William C. Cheng

RAID Level 1

12

Copyright © William C. Cheng

Striping Unit Size

The above two cases have *same transfer time* = can *reduce seek time* by using a *larger striping unit*

9

Copyright © William C. Cheng

Striping: The Effective Disk

- Improved effective transfer speed
- parallelism
- No improvement in seek and rotational delays
- sometimes worse
- A system depending on N disks is much more likely to fail than one depending on one disk
- if probability of one disk failing is f
- probability of N -disk system failing is $(1 - (1 - f)^N)$
- assumes failures are i.i.d., which is probably wrong ...
- i.i.d.*: independent and identically distributed

10

Copyright © William C. Cheng

How To Stripe?

- How to stripe?
- what's the best *stripe width* (i.e., across how many disks)?
- how large should be the *striping unit* (i.e., how much to put on one disk before moving to the next disk)?
- Concurrency Factor: how many requests are available to be executed at once?
- one request in queue at a time
- concurrency factor = 1
- e.g., one single-threaded application placing one request at a time
- many requests in queue
- concurrency factor > 1
- e.g., multiple threads placing file-system requests
- the larger the concurrency factor, the less important striping is
- in general, performance is better with *larger striping unit*


7

Copyright © William C. Cheng

Striping Unit Size

Bottom solution seems better = data transfer time is 1/4 of the solution on the top


8


17


Copyright © William C. Cheng

RAID 4 VS. RAID 5

- ↳ Lots of small writes
- ↳ RAID 5 is best
- ↳ Mostly large writes
 - ↳ multiples of stripes
 - ↳ either is fine
- ↳ Expansion
 - ↳ add an additional disk or two
 - ↳ RAID 4: add them and recompute parity
 - ↳ RAID 5: add them, recompute parity, shuffle data blocks among all disks to reestablish check-block pattern
- ↳ Write performance
 - ↳ RAID 4: parity disk have workload multiple of other disks
 - ↳ RAID 5: same workload on all disks on the average
- ↳ One disk failure
 - ↳ RAID 4: parity disk have workload multiple of other disks
 - ↳ RAID 5: work load spread out more evenly


17


Operating Systems - CSCI 402


16


Copyright © William C. Cheng

Beyond RAID 5

- ↳ RAID 6
 - ↳ like RAID 5, but additional parity
 - ↳ handles two failures
- ↳ Cascaded RAID
 - ↳ RAID 1+0 (RAID 10)
 - ↳ striping across mirrored drives
 - ↳ RAID 0+1
 - ↳ two striped sets, mirroring each other

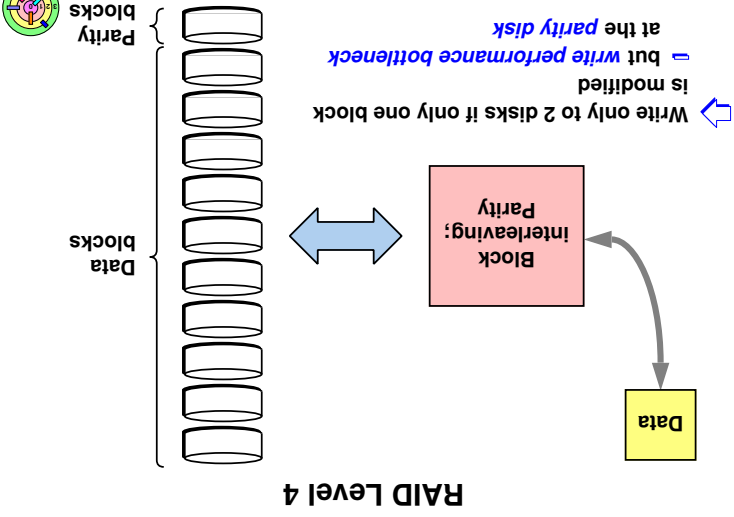

16

Operating Systems - CSCI 402



15

Copyright © William C. Cheng


RAID Level 4



Write only to 2 disks if only one block is modified but write performance bottleneck at the parity disk

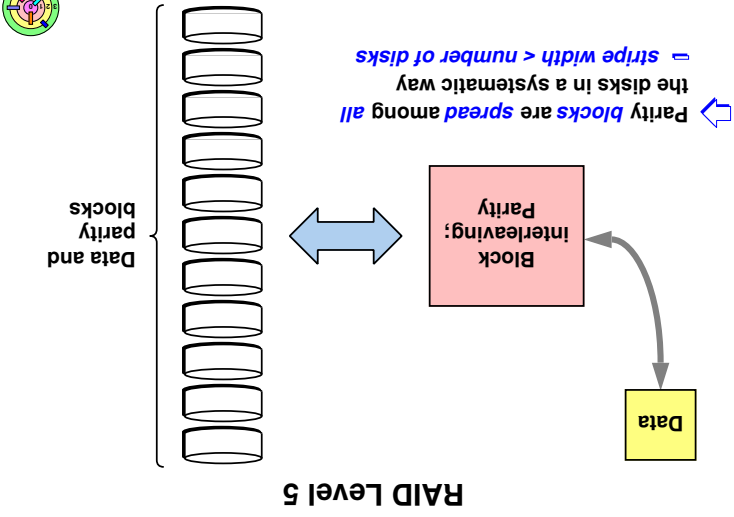

15

Operating Systems - CSCI 402



16

Copyright © William C. Cheng


RAID Level 5



Parity blocks are spread among all the disks in a systematic way stripe width < number of disks

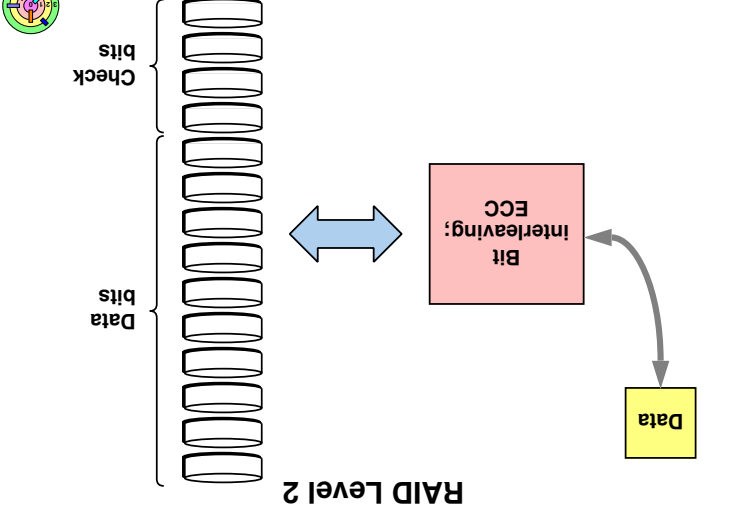

16


Operating Systems - CSCI 402


13


Copyright © William C. Cheng

RAID Level 2



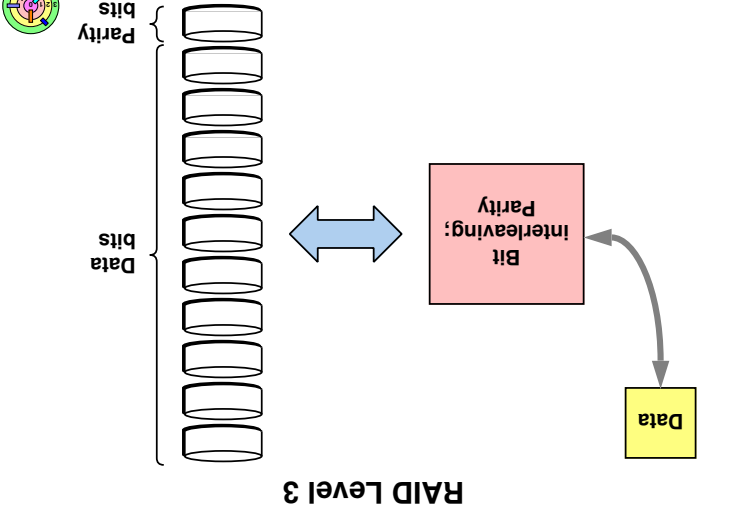

13


Operating Systems - CSCI 402


14

Copyright © William C. Cheng

RAID Level 3




14

Operating Systems - CSCI 402

6.5 Flash Memory

- Flash Technology
- Flash-Aware File Systems
- Augmenting Disk Storage



19

Copyright © William C. Cheng

Operating Systems - CSCI 402

Beyond Disks: Flash

- Pro
 - Flash block \approx file-system block
 - Random access
 - no seek, no rotational latency
 - Low power
 - Vibration-resistant
- Con
 - Limited lifetime
 - Write is expensive
 - Cost more than disks
 - 128GB SSD: ~\$300
 - 1TB disk: ~\$60



20

Copyright © William C. Cheng

Operating Systems - CSCI 402

Flash Memory

- Two technologies
 - NOR
 - byte addressable
 - NAND
 - page addressable (about 1-4KB per page and 512KB per block)
 - cheaper
 - suitable for file systems use
 - limit on P/E (program/erase) cycle, about 10,000
- Writing
 - newly "erased" block is all ones
 - "programming" changes some ones to zeroes
 - per byte in NOR; per page in NAND (multiple pages/block)
 - to change zeroes to ones, must erase entire block
 - can erase no more than $\sim 100k$ times/block



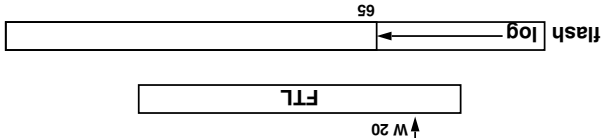
21

Copyright © William C. Cheng

Operating Systems - CSCI 402

Coping

- Wear leveling
 - spread writes (erasures) across entire drive
 - approache:
 - flash translation layer (FTL)
 - log-structured file system
 - blocks on the flash drive are used sequentially
- FTL: Flash translation layer (often on a separate device)
 - specification from 1994
 - provides disk-like block interface (firmware on device controller)
 - maps disk blocks to flash blocks
 - mapping changed dynamically to effect wear-leveling



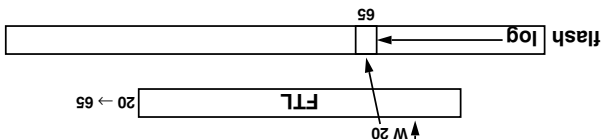
22

Copyright © William C. Cheng

Operating Systems - CSCI 402

Coping

- Wear leveling
 - spread writes (erasures) across entire drive
 - approaches:
 - flash translation layer (FTL)
 - log-structured file system
 - blocks on the flash drive are used sequentially
- FTL: Flash translation layer (often on a separate device)
 - specification from 1994
 - provides disk-like block interface (firmware on device controller)
 - maps disk blocks to flash blocks
 - mapping changed dynamically to effect wear-leveling



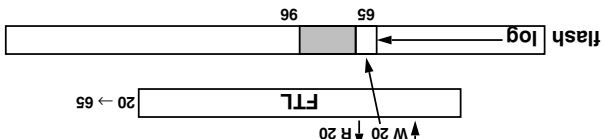
23

Copyright © William C. Cheng

Operating Systems - CSCI 402

Coping

- Wear leveling
 - spread writes (erasures) across entire drive
 - approaches:
 - flash translation layer (FTL)
 - log-structured file system
 - blocks on the flash drive are used sequentially
- FTL: Flash translation layer (often on a separate device)
 - specification from 1994
 - provides disk-like block interface (firmware on device controller)
 - maps disk blocks to flash blocks
 - mapping changed dynamically to effect wear-leveling



24

Copyright © William C. Cheng

Operating Systems - CSCI 402

30

Copyright © William C. Cheng

Flash without FTL

Known as memory technology device (MTD)

software wear-leveling

perhaps other tricks

Operating Systems - CSCI 402

29

Copyright © William C. Cheng

Flash with FTL

Which file system?

FAT32 (sort of like SFS, but from Microsoft)

NTFS

FFS

Ext3

All were designed to exploit disks

much of what they do are irrelevant for flash

Operating Systems - CSCI 402

28

Copyright © William C. Cheng

Coping

Wear leveling

spread writes (erasures) across entire drive

approaches:

flash translation layer (FTL)

log-structured file system

blocks on the flash drive are used sequentially

FTL: Flash translation layer (often on a separate device)

specification from 1994

provides disk-like block interface (firmware on device controller)

maps disk blocks to flash blocks

mapping changed dynamically to effect wear-leveling

Operating Systems - CSCI 402

27

Copyright © William C. Cheng

Coping

Wear leveling

spread writes (erasures) across entire drive

approaches:

flash translation layer (FTL)

log-structured file system

blocks on the flash drive are used sequentially

FTL: Flash translation layer (often on a separate device)

specification from 1994

provides disk-like block interface (firmware on device controller)

maps disk blocks to flash blocks

mapping changed dynamically to effect wear-leveling

Operating Systems - CSCI 402

26

Copyright © William C. Cheng

Coping

Wear leveling

spread writes (erasures) across entire drive

approaches:

flash translation layer (FTL)

log-structured file system

blocks on the flash drive are used sequentially

FTL: Flash translation layer (often on a separate device)

specification from 1994

provides disk-like block interface (firmware on device controller)

maps disk blocks to flash blocks

mapping changed dynamically to effect wear-leveling

Operating Systems - CSCI 402

25

Copyright © William C. Cheng

Coping

Wear leveling

spread writes (erasures) across entire drive

approaches:

flash translation layer (FTL)

log-structured file system

blocks on the flash drive are used sequentially

FTL: Flash translation layer (often on a separate device)


specification from 1994

provides disk-like block interface (firmware on device controller)

maps disk blocks to flash blocks

mapping changed dynamically to effect wear-leveling

Operating Systems - CSCI 402



35

Copyright © William C. Cheng

Linux

- Linux had 57 file systems built for it to date!
- FFS
 - ext2
 - ext3 (journaling - crash resiliency)
 - ReiserFS (B-tree everywhere)
 - ext4
 - extents (optimize read/write)
 - LVM
 - hash trees for directories
- Btrfs (Oracle)

Operating Systems - CSCI 402



36

Copyright © William C. Cheng

Windows NT

- NTFS
 - extents (optimize read/write)
 - B-trees (optimize directory lookup)
 - journaling (crash resiliency)
- Mac OS X
 - HFS+ (planned to use ZFS but dropped the idea)
 - extents (optimize read/write)
 - B*-trees (optimize directory lookup)
 - journaling (crash resiliency)

Operating Systems - CSCI 402



33

Copyright © William C. Cheng

Flash as Part of the Hierarchy

- Flash as log device
 - aggregate write throughput sufficient, but latency is bad
 - augment with DRAM and a "super-capacitor"
- Flash as cache
 - large level-2 cache
 - integrated into ZFS
 - can use cheaper (slower) disks with no loss of performance
 - reduced power consumption

Operating Systems - CSCI 402



34

Copyright © William C. Cheng

6.6 Case Studies

- FFS
- Ext3
- Reiser FS
- NTFS
- WAF
- ZFS

Operating Systems - CSCI 402



31

Copyright © William C. Cheng

JFFS and JFFS2

- Journaling flash file system
 - log-based: no journal!
 - each log entry contains inode info and some data
 - garbage collection copies info out of partially obsolete blocks, allowing block to be erased
 - complete index of inodes (i.e., meta-data) kept in RAM
 - entire file system must be read when mounted

Operating Systems - CSCI 402

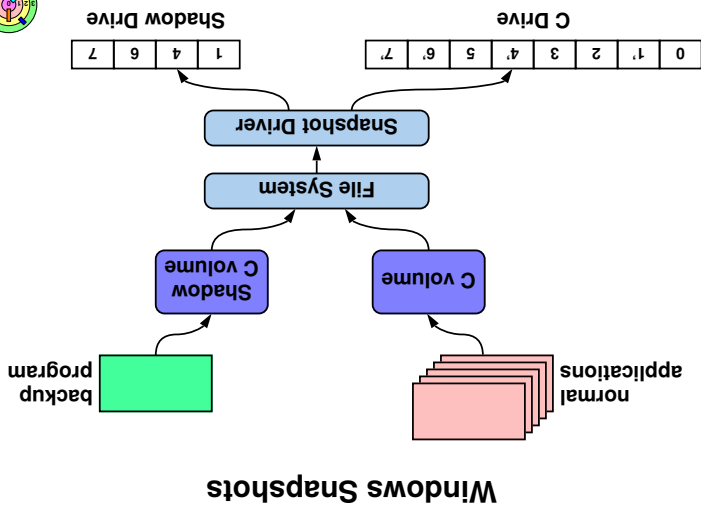

32

Copyright © William C. Cheng

UBI / UBIFS

- UBI (unsorted block images)
 - supports multiple logical volumes on one flash device
 - performs wear-leveling across entire device
 - handles bad blocks
- UBIFS
 - file system layered on UBI
 - it really has a journal (originally called JFFS3)
 - file map kept in flash as B+ tree
 - no need to scan entire file system when mounted

Operating Systems - CSCI 402



Backups

- Want to back up a file system
 - while still using it
 - files are being modified while the backup takes place
 - applications may be in progress - files in inconsistent states
- Solution
- have critical applications quickly reach a safe point and pause
 - snapshot the file system
 - resume applications
 - back up the snapshot

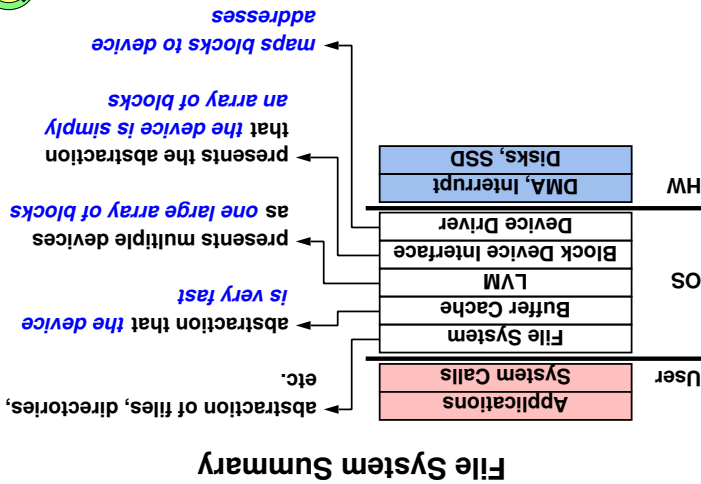
Extra Slides

NTFS

- "Volume aggregation" options
- spanned volumes
- RAID 0 (striping)
- RAID 1 (mirroring)
- RAID 5
- snapshots

Journaling

- Why did everyone choose journaling and not shadow pages?
- Journaling can be added to any existing file system



47

Copyright © William C. Cheng

Consistency Points ... and Beyond

Consistency points taken every ~10 seconds

too relaxed for many applications

NFS

databases

Solution ...

battery-backed up RAM

a.k.a. non-volatile RAM (NVRAM)

46

Copyright © William C. Cheng

Snapshots

Periodic snapshots kept of file system

made easy with shadow paging

45

Copyright © William C. Cheng

WAFL

Runs on special-purpose OS

machine is dedicated to being a *filer*

handles both NFS and CIFS requests

Utilizes shadow paging and log-structured writes

Provides snapshots

46

Copyright © William C. Cheng

WAFL and RAID

consistency point

blocks

Parity blocks

43

Copyright © William C. Cheng

NTFS File Records

Name

Standard attributes

Object ID

Properties

Data stream

Name

Standard attributes

Object ID

Data stream

Extent 1

Extent 2

Extent 3

44

Copyright © William C. Cheng

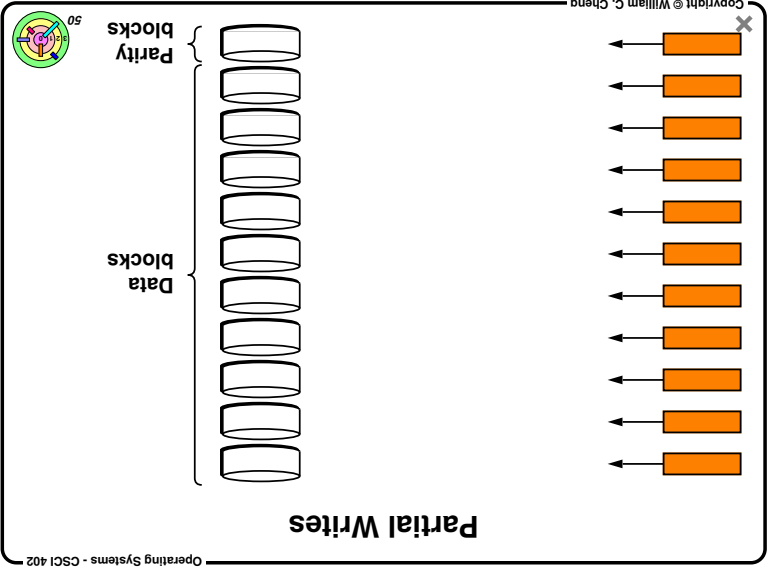
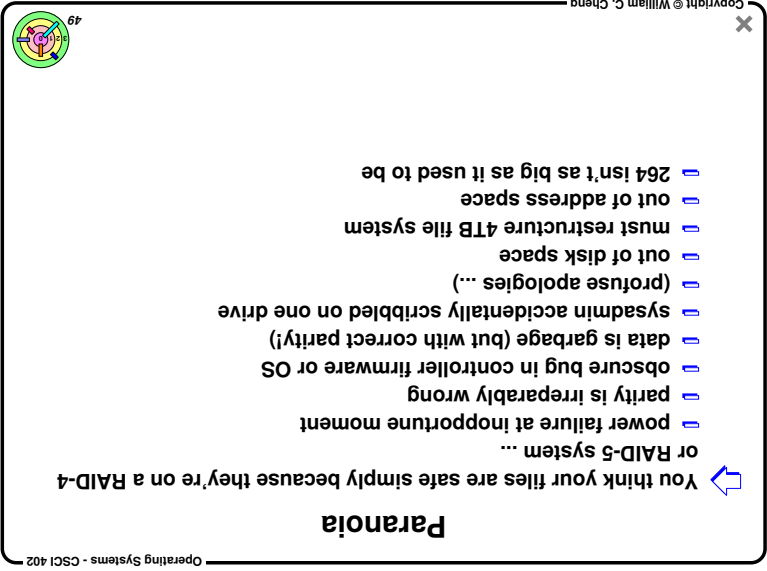
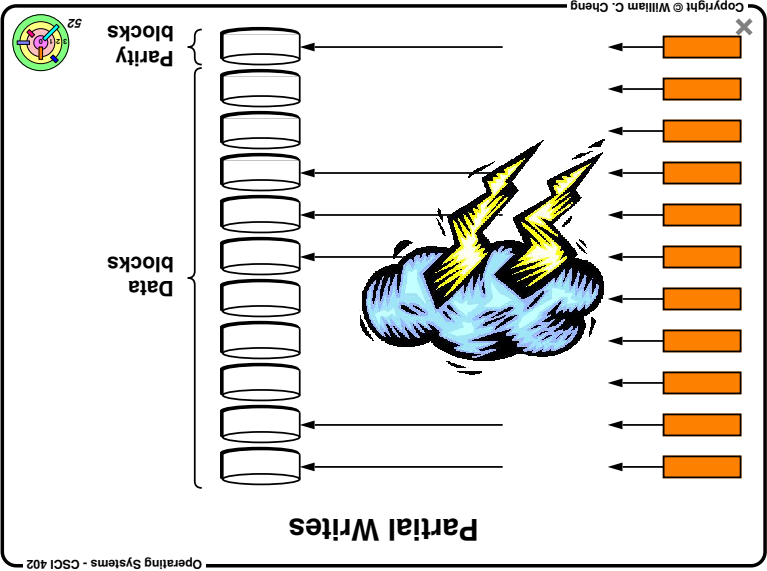
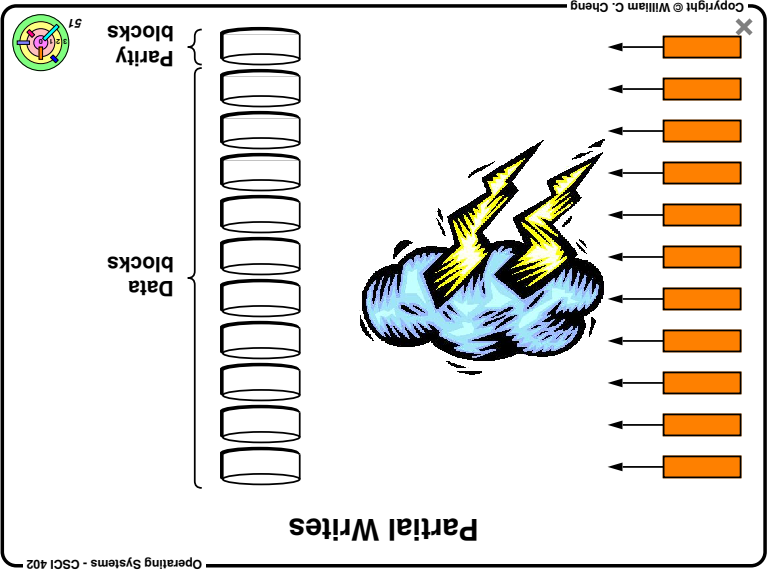
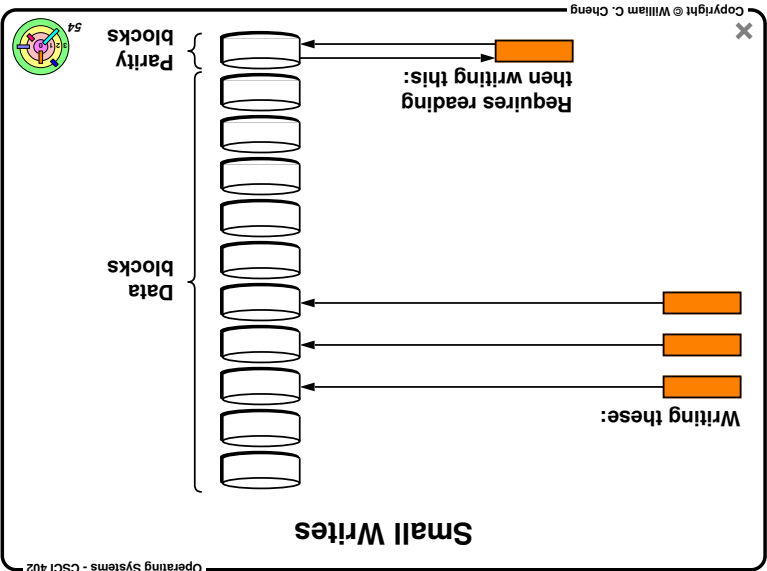
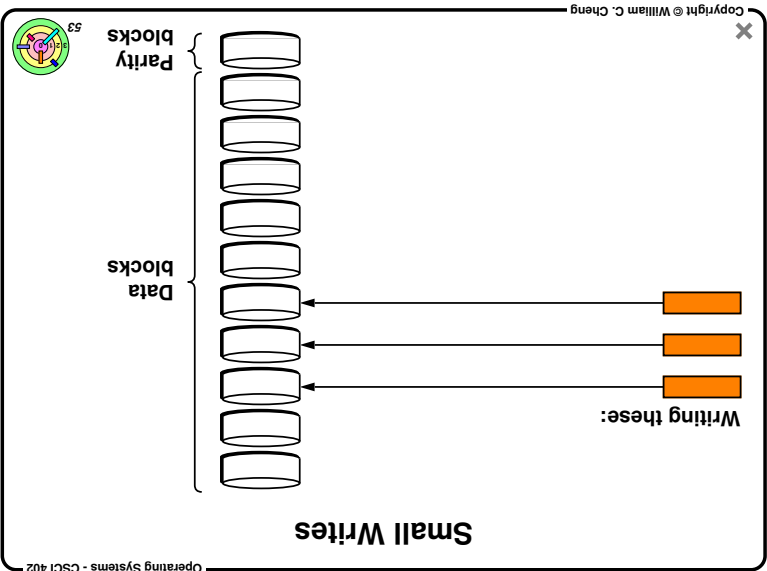
Additional NTFS Features

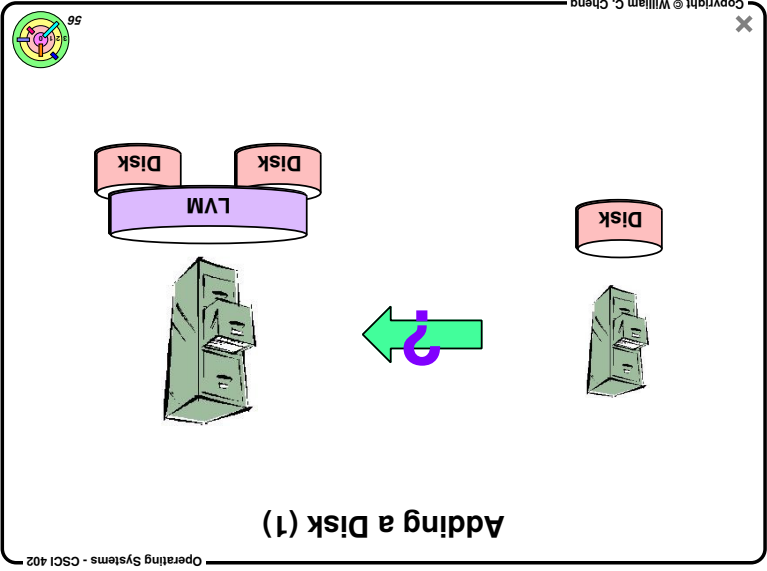
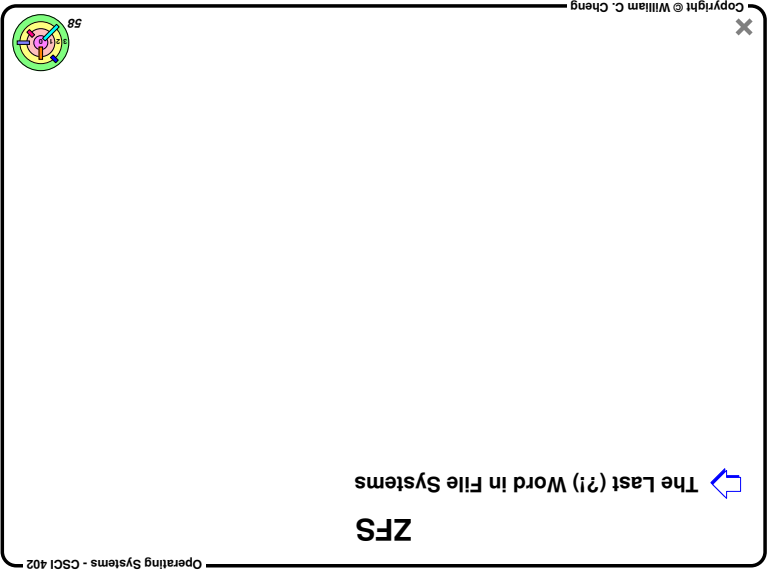
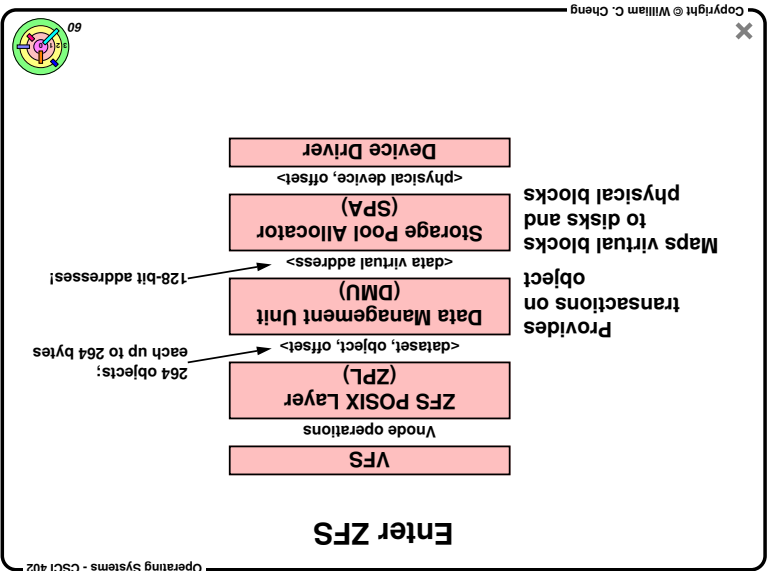
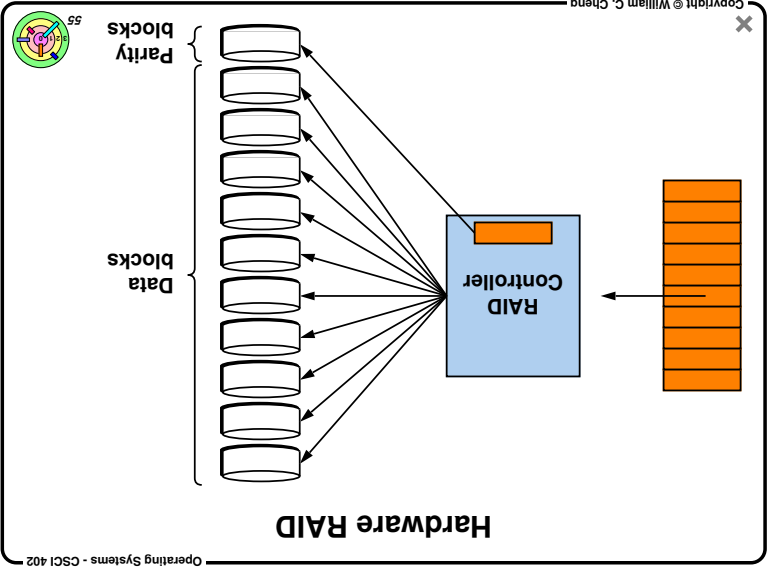
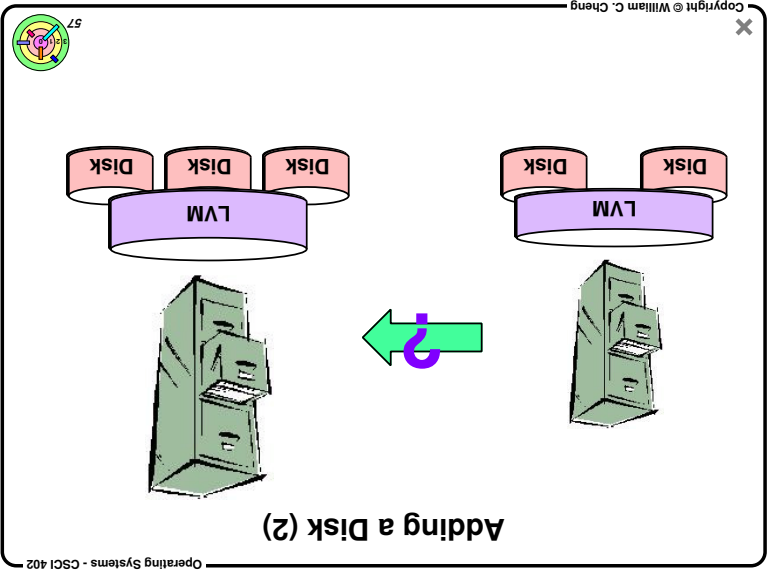
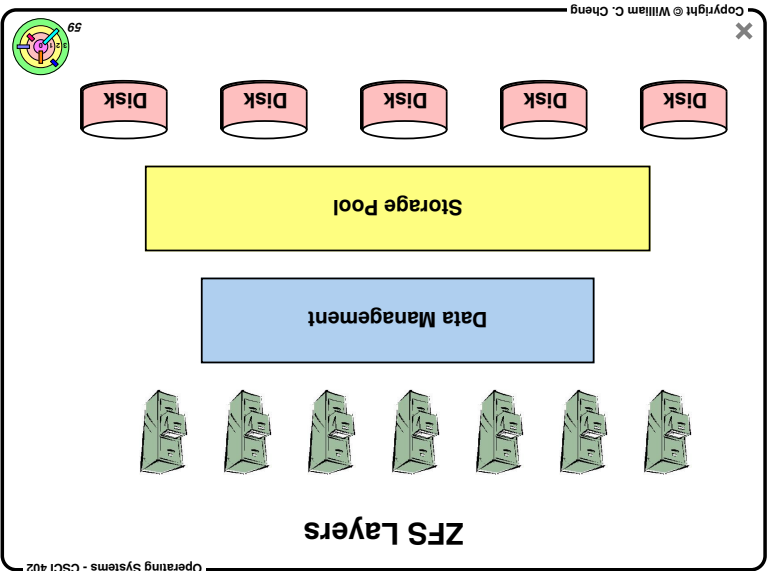
Data compression

run-length encoding of zeroes

compressed blocks

Encrypted files





65

Scenarios

- Power failure at inopportune moment
- "live data" is not modified
- single lost write can be recovered
- Obscure bug in controller firmware or OS
- detected by checksum in pointer
- Sysadmin accidentally scribbled on one drive
- detected and repaired
- Out of disk space
- add to the pool; SPA will cope
- Out of address space
- 2128 is big
- 1 address per cubic yard of a sphere bounded by the orbit of Neptune

63

RAID-Z

Software Dynamic Striping

66

And There's More ...

- Adaptive replacement cache
- Advanced prefetching

64

RAID-Z

Adding a Disk

61

Shadow-Page Tree (with a twist ...)

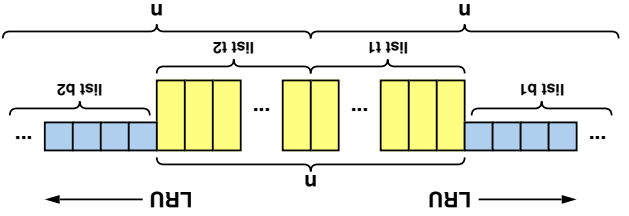
62

Storage Pool Allocator

Data Management Unit (DMU)

Storage Pool Allocator (SPA)

Mirroring, or RAID

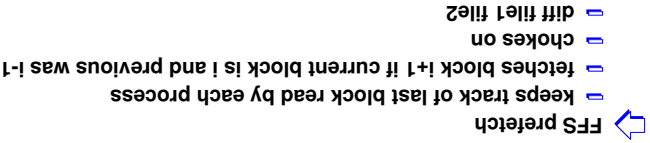


Adaptive Replacement Cache

- ```

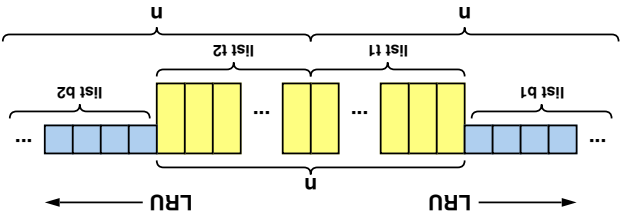
cache hit:
 if in t1 or t2, block becomes MRU(t2)
 otherwise
 if block is referred to by b1, increase t1 space at expense of t2
 otherwise
 increase t2 space at expense of t1
 if t1 is full, evict LRU(t1) and make it MRU(b1)
 if t2 is full, evict LRU(t2) and make it MRU(b2)
 insert block as MRU(t2)

```



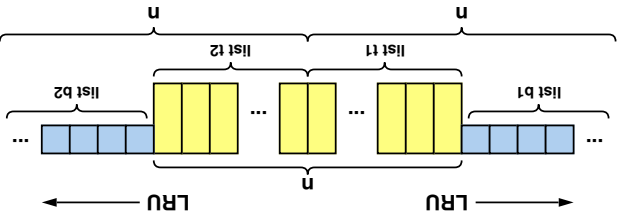
## Prefetch

- keeps track of last block read by each process
- fetches block  $i+1$  if current block is  $i$  and previous was  $i-1$
- chokes on
- diff file1 file2



## Adaptive Replacement Cache

- $t1$ : LRU list of blocks referenced once
- $t1$  list (most recently used) contain contents
- $t1$  list (least recently used) contain just references
- $t2$ : LRU list of blocks referenced more than once
- $t2$  list (most recently used) contain contents
- $t2$  list (least recently used) contain just references



## Adaptive Replacement Cache

- cache miss:
  - if t1 is full
  - evict LRU(t1) and make it MRU(b1)
  - referenced block becomes MRU(t1)



## LRU Caching

- LRU cache holds n least-recently-used disk blocks
- working sets of current processes
- New process reads n-block file sequentially
  - cache fills with this file's blocks
  - old contents flushed
  - new cache contents never accessed again



### (Non-Adaptive) Solution

- half of it is for blocks that have been referenced exactly once
  - half of it is for blocks that have been referenced more than once
- is 50/50 split the right thing to do?

×

Copyright © William C. Cheng

73

zfetchn

Tracks multiple prefetch streams

Handles four patterns

forward sequential access

backward sequential access

forward strided access

backward strided access

iterating across columns of matrix stored by columns

backward strided access

