Introduction to COMs

Alan G. Yoder, Ph.D.
SNIA Technical Council
Huawei Technologies, LLC

Outline

- COM overview
- How they work

COM overview

- COM = Capacity Optimization Method
- Basic idea: figure out a way to store more data in less space
  - energy use is theoretically proportional to space used

Currently acknowledged COMs

- Parity RAID
- Thin Provisioning
- Read-only Delta Snapshots
- Writeable Delta Snapshots
- Data Deduplication
- Compression

RAID history

- Mainframe era
  - Disk drives looked like washing machines
  - Big and very reliable
- Idea
  - Mass together a bunch of small cheap drives and add a fault tolerance scheme, to make storage cheaper ➔
- **R**edundant **A**rray of **I**nexpensive **D**isks
  - Really was a better storage solution (failures are cheaper)
  - But OOOPSIE, turns out we need “Enterprise Class” drives ($$$)
- **R**edundant **A**rray of **I**ndependent **D**isks

RAID is for data protection

- The problem: disks break (usually invisibly)

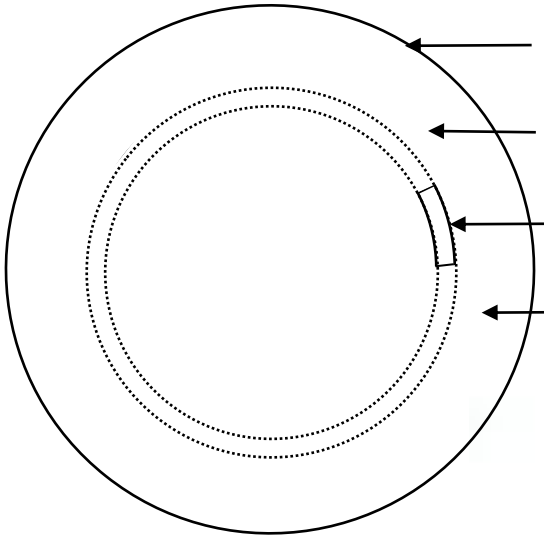
- They also lie, cheat and steal

All of these issues cause *data loss*. Preventing that is called *data protection*.

To prevent data loss, we have to have another copy of the data, or something that the lost data can be reconstructed from.

RAID – disk organization

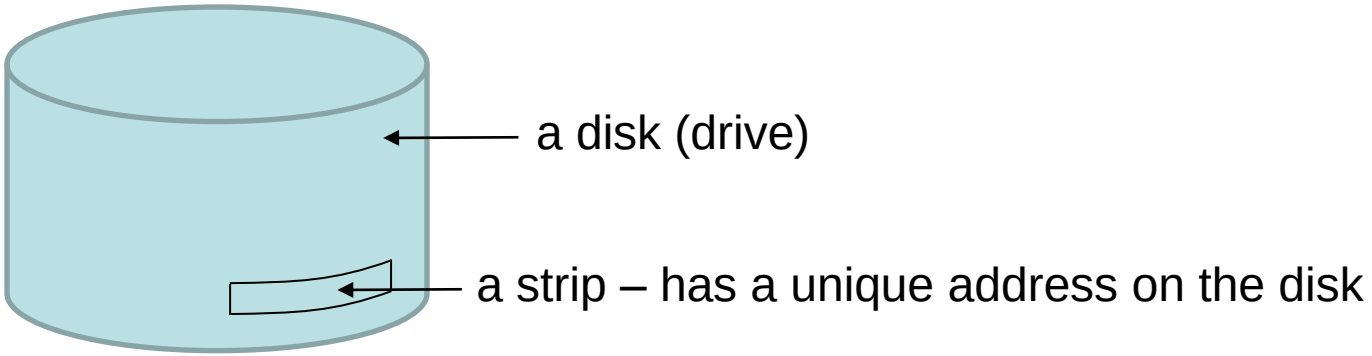
(not to scale)



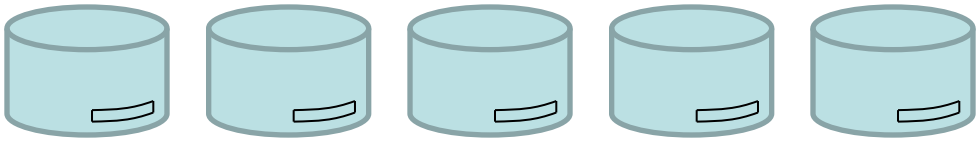
RAID – disk organization

Now, same thing using the traditional symbol for a disk drive

(it's easier to draw)



Stripe together all the strips in a RAID Group that have the same address



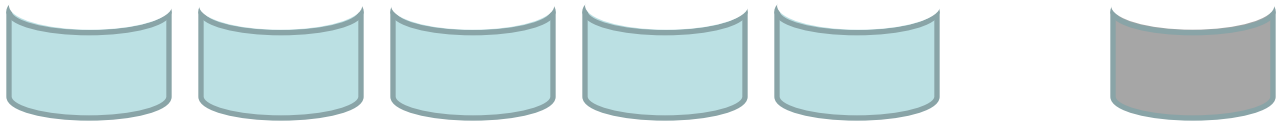
Now is the ti	me for all good	men to come to	the aid of their p	arty

This is a RAID stripe.

Storing data this way is called *striping*

RAID – erasure codes

- "Erasure codes"
  - Coding schemes that tolerate a certain amount of accidental content erasure
  - Reed-Solomon, XOR, others
- Simplest is XOR (works like ordinary addition)
- Example (traditional RAID) – tolerates one erasure
  - we add a *parity* strip somewhere

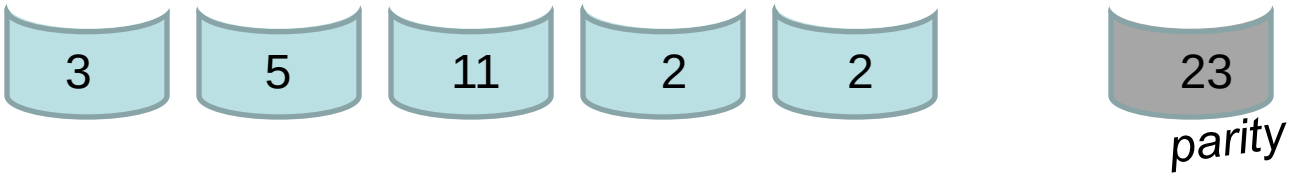


our strips, assembled into a stripe

parity

RAID – erasure codes

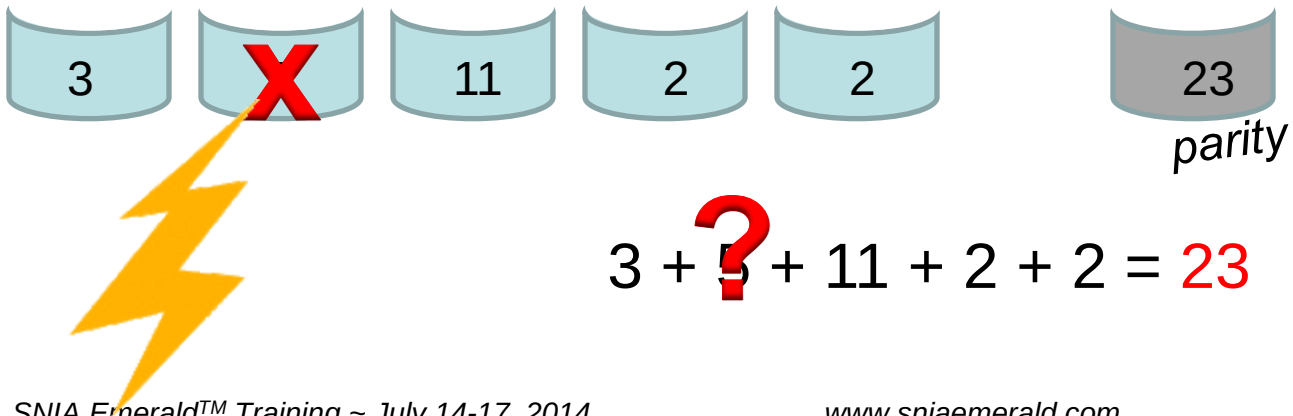
- "Erasure codes"
  - Coding schemes that tolerate a certain amount of accidental content erasure
  - Reed-Solomon, XOR, others
- Simplest is XOR (works like ordinary addition)
- Example (traditional RAID)
  - Add together the values in the strips, and that's our parity



3 + 5 + 11 + 2 + 2 = 23

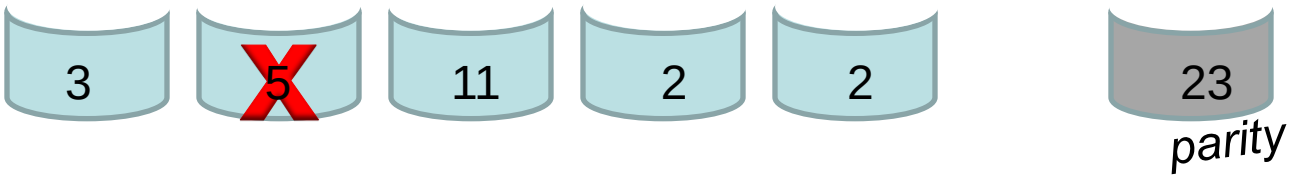
RAID – erasure codes, cont.

- "Erasure codes"
  - Coding schemes that tolerate a certain amount of accidental content erasure
  - Reed-Solomon, XOR, others
- Simplest is XOR (works like ordinary addition)
- Example (traditional RAID)
  - when something goes pszzzft….



RAID – erasure codes, cont.

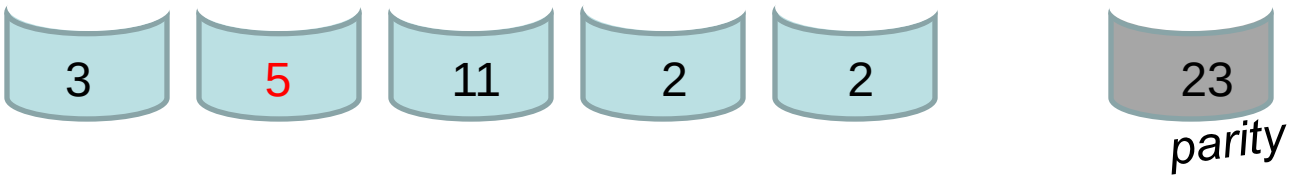
- "Erasure codes"
  - Coding schemes that tolerate a certain amount of accidental content erasure
  - Reed-Solomon, XOR, others
- Simplest is XOR (works like ordinary addition)
- Example (traditional RAID)
  - we *recalculate parity*



23 – 3 – 11 – 2 – 2 = 5

RAID – erasure codes, cont.

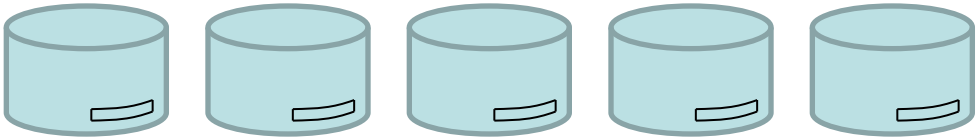
- "Erasure codes"
  - Coding schemes that tolerate a certain amount of accidental content erasure
  - Reed-Solomon, XOR, others
- Simplest is XOR (works like ordinary addition)
- Example (traditional RAID)
  - and *voila!*



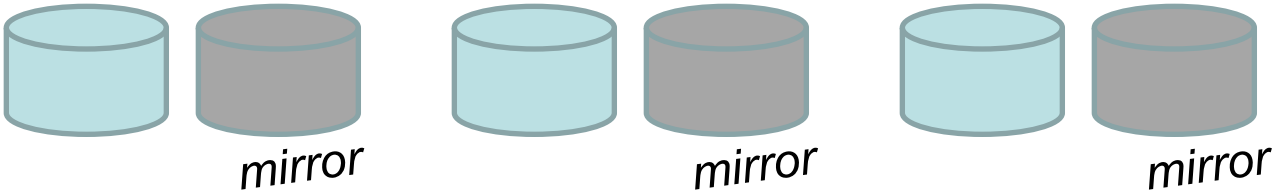
$$23 - 3 - 11 - 2 - 2 = 5$$

RAID types

- RAID 0 = striping (AID, not really RAID)



- RAID 1 = mirroring – the gold standard for enterprise data protection



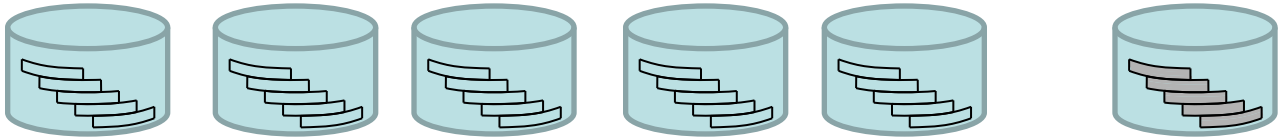
  - 100% space overhead
  - some performance gain

RAID types, cont.

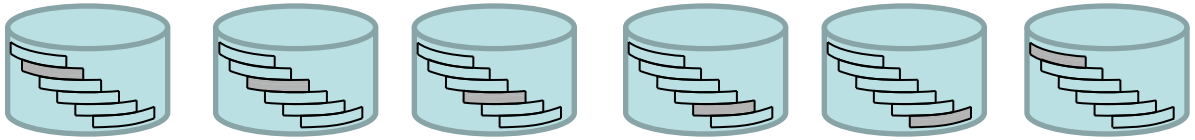
- RAID 2 and 3

  no longer used

- RAID 4 – dedicated parity drive



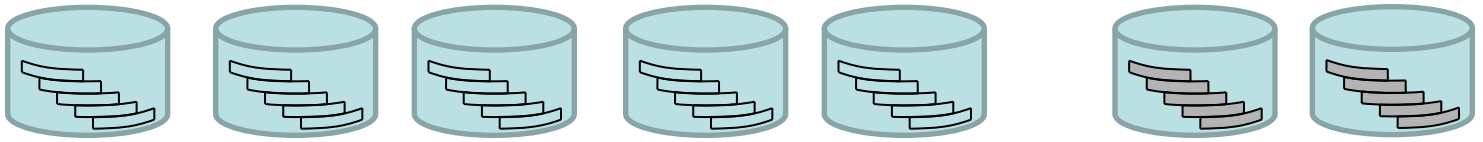
- RAID 5 –parity striped across RAID set



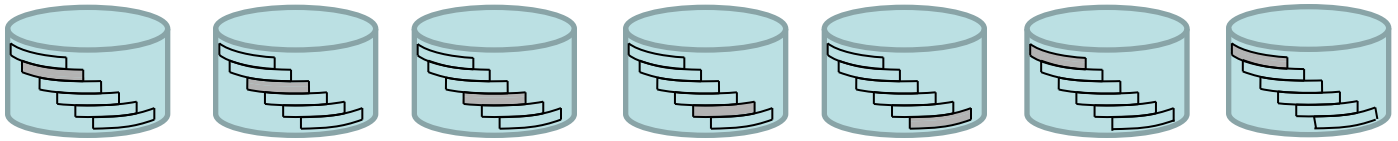
"virtual" parity drive

RAID types, cont.

- RAID 6 – two parity strips per stripe



or



- tolerates failure of any two disks in a group

RAID types, cont.

- RAID 10 (a.k.a. RAID 1+0)
  - mirrored disks striped together into a group
- RAID 0+1
  - two RAID 0 stripes mirrored together
- RAID 7 etc.
  - OMG, WTF, how many of these are there?
    - too many

The key to it all

- For ENERGY STAR, all that matters is the ratio of data segments to parity segments
  - A larger number translates directly into power savings
  - Less power to store a given amount of data is the goal
- RAID 1 is 1 to 1
  - This is what we want to improve on
  - E.g. RAID 5 with 8 drives is 7 to 1

RAID summary

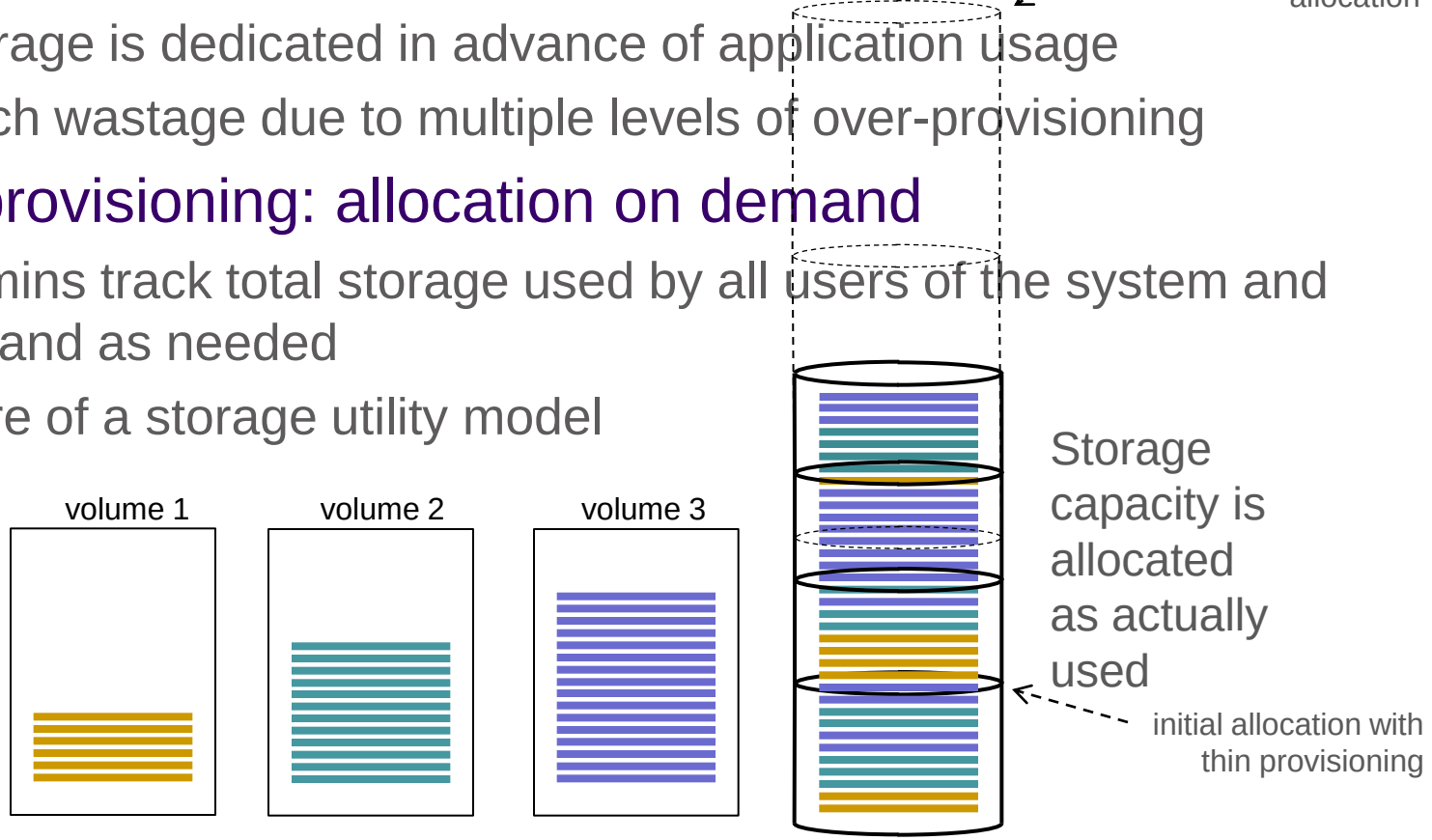
Types of RAID

RAID 0	simple striping	not really RAID	
RAID 1	mirroring	NOT parity RAID	
RAID 4	parity on a separate drive	okay for ES	*only good for smaller drives*
RAID 5	parity striped across drives	okay for ES	
RAID 6	double parity	okay for ES	*protection against failures during RAID reconstruct*
"erasure codes"	non-XOR parity	okay for ES	
distributed parity	multiple parity, widely distributed [1]	okay for ES	
RAID 0+1, 1+0, RAID 10	striping+mirroring	NOT parity RAID	
replication	e.g. Hadoop, AWS	NOT parity RAID	

[1] Usually in "RAID groups" of 16

Thin Provisioning

- Traditional: pre-allocation of storage space
  - Storage is dedicated in advance of application usage
  - Much wastage due to multiple levels of over-provisioning
- Thin provisioning: allocation on demand
  - Admins track total storage used by all users of the system and expand as needed
  - More of a storage utility model

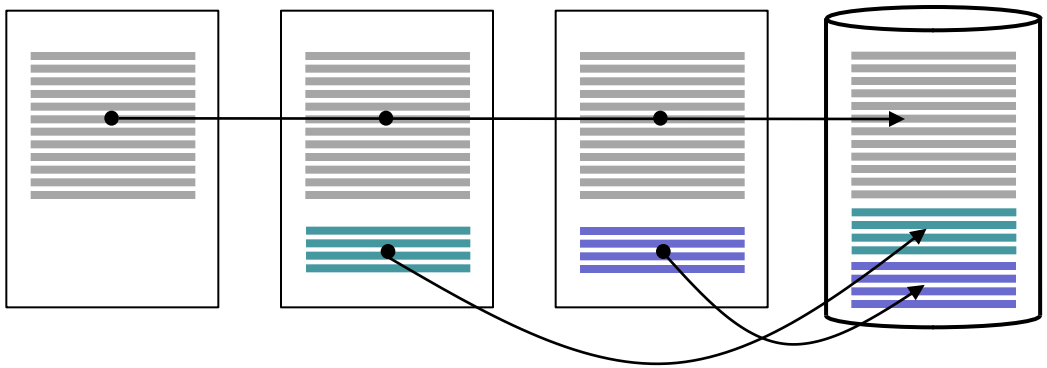


Thin provisioning, cont.

- Thin provisioning power saving effects are indirect
  - Avoid buying and powering up storage ahead of need
  - I.e. minimize the amount of unused space on a system
  - Good administrative practices greatly increase its effectiveness
    - An empty system is an empty system, no matter what

Delta Snapshots

- Snapshot
  - A Point-in-time (PIT) copy of some data
  - Usually at a volume or filesystem level
- Traditional method
  - Full copy
  - Lock volume, suspend or log writes, make copy, write log, unlock
- Delta method
  - Copy on write
  - Snapshots share blocks



Delta Snapshots cont.

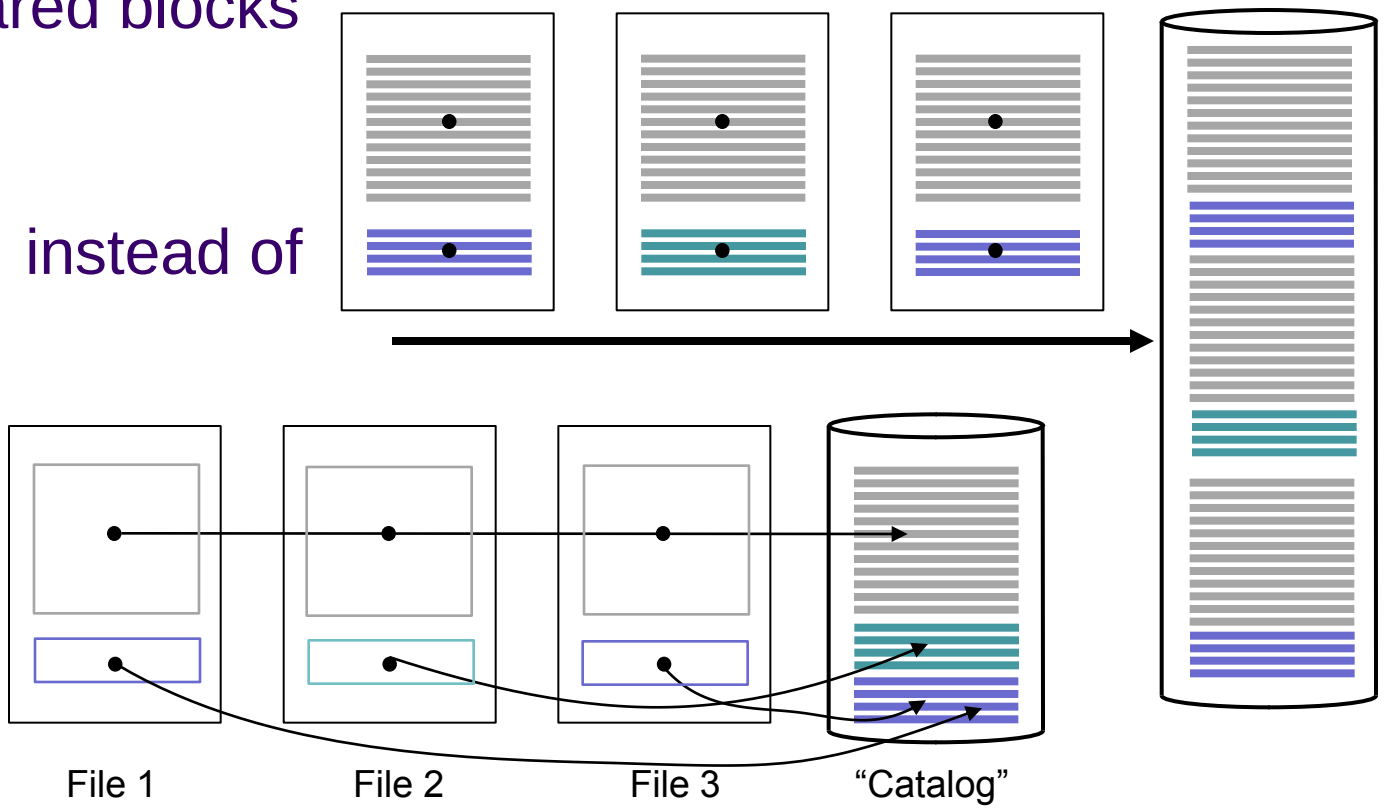
- Read only
  - Live copy continues as before
  - PIT copy cannot be written
  - Useful for backups
- Writeable
  - Live copy continues as before
  - PIT copy can be written
  - Useful for “what if” scenarios, test runs on live data
  - Example: NetApp “flexclones”

Data Deduplication

- A.K.A. “dedup”
- Basic idea: replace duplicate blocks with pointers to shared blocks

instead of

do



Data Deduplication

- Two fundamental types
  - "Inline" – Dedup at wire speed, before writing to disk
    - Usually used for streaming backup systems
    - Note: streaming backup systems are not "online" systems in the SNIA taxonomy, so are not covered by the ES spec
  - "Post process" – Dedup performed after initial write to non-volatile media
- Global vs. local
  - Global dedup works across all the nodes in a cluster– very hard
    - i.e. system-wide, not global in the planetary sense
  - Most dedup is "local" to a given node

Deduplication, cont.

- Many variations
  - File or object level
    - Coarsest grained, least overhead
  - Block level
    - Granularity at 4K or larger
  - Variable-size
    - Finer granularity, but more overhead
  - Yada yada
    - Whatever

Compression

- Old and venerable technology
- Well understood
- Zip, pkzip, WinRAR, others
- Finer grained than dedup
  - Byte level dedup inside typically a 32K sliding window
- Difficult, but possible, to combine with dedup
- Which is “better” depends on dataset

COM Benefits summary

- All COMs allow you to store more data in less space
- Parity RAID
  - replacement for mirroring
  - Usually 40-some percent power savings over RAID 1
- Thin provisioning
  - Can take systems from 30% utilization (legacy) to 80%
- Delta snapshots
  - Huge savings possible when change delta is small
- Deduplication
  - Savings depend on several factors, can be large
  - Think of backing up thousands of laptops, all originally burned from the same master image
- Compression
  - Savings vary with data characteristics, can be large
  - As compression is local to a file or block, it can't achieve what dedup can

Q & A