AutoRAID

Goals: automate the efficient replication of data in a RAID

- RAIDs are hard to setup and optimize
- Mix fast mirroring (2 copies) with slower, more space-efficient parity disks
- Automate the migration between these two levels

Levels of RAID (those in **bold** are actually used):

- RAID 0: striping with no parity (just bandwidth)
- **RAID 1: Mirroring** (simple, fast, but requires 2x storage)
  - Reads faster, writes slower (why?)
- RAID 2: bit interleaving with error-correcting codes (ECC)
- **Dedicated parity disk (RAID level 3), byte-level striping**
  - dedicated parity disk is a write bottleneck, since every write also writes parity
- RAID 4:dedicated parity disk, block-level striping
- **RAID 5: Rotating parity disk, block-level striping**
  - most popular; rotating disk spreads out parity load
- RAID 6: RAID 5 with two parity blocks (tolerates two failures)

RAID small-write problem:

- to overwrite part of a block required 2 reads and 2 writes!
- read data, read parity, write data, write parity

Each kind of replication has a narrow range of workloads for which it is best...

- Mistake ⇒ 1) poor performance, 2) changing layout is expensive and error prone
- Also difficult to add storage: new disk ⇒ change layout and rearrange data...

(another problem: spare disks are wasted)

Key idea: mirror active data (hot), RAID 5 for cold data

- Assumes only part of data in active use at one time
- Working set changes slowly (to allow migration)

Where to deploy:

- sys-admin: make a human move around the files.... BAD. painful and error prone
- File system: best choice, but hard to implement/deploy; can't work with existing systems

AutoRAID

- Smart array controller: (magic disk) block-level device interface. Easy to deploy because there is a well-defined abstraction; enables easy use of NVRAM (why?)

Features:

- Block Map: level of indirection so that blocks can be moved around among the disks
  - implies you only need one "zero block" (all zeroes), a variation of copy on write
  - in fact could generalize this to have one real block for each unique block
- Mirroring of active blocks
- RAID 5 for inactive blocks or large sequential writes (why?)
- Start out fully mirrored, then move to 10% mirrored as disks fill
- Promote/demote in 64K chunks (8-16 blocks)
- Hot swap disks, etc. (A hot swap is just a controlled failure.)
- Add storage easily (goes into the mirror pool)
  - useful to allow different size disks (why?)
- No need for an active hot spare (per se); just keep enough working space around
- Log-structured RAID 5 writes. (Why is this the right thing? Nice big streams, no need to read old parity for partial writes)

Sizes:

- PEX = 1MB
- PEG = set of PEXs
- segment = 128K
- relocation block = RB = 64K (size for holeplugging)

Issues:

- When to demote? When there is too much mirrored storage (>10%)
- Demotion leaves a hole (64KB). What happens to it? Moved to free list and reused
- Demoted RBs are written to the RAID5 log, one write for data, a second for parity
- Why log RAID5 better than update in place? Update of data requires reading all the old data to recalculate parity. Log ignores old data (which becomes garbage) and writes only new data/parity stripes.
- When to promote? When a RAID5 block is written... Just write it to mirrored and the old version becomes garbage.
- How big should an RB be? Bigger $\Rightarrow$ finer-grain migration, smaller $\Rightarrow$ less mapping information, bigger $\Rightarrow$ fewer seeks
- How do you find where an RB is? Convert addresses to (LUN, offset) and then lookup RB in a table from this pair. Map size = Number of RBs and must be proportional to size of total storage.
- Controller uses cache for reads
- Controller uses NVRAM for fast commit, then moves data to disks. What if NVRAM is full? Block until NVRAM blocks flushed to disk, then write to NVRAM.

AutoRAID

- Disks writes normally go to two disks (since newly written data is “hot”). Must wait for both to complete (why?). Does the host have to wait for both? No, just for NVRAM.
- What happens in the background? 1) compaction, 2) migration, 3) balancing.
- Compaction: clean the RAID5 and plug holes in the mirrored disks. Do mirrored disks get cleaned? Yes, when a PEG is needed for RAID5; i.e., pick a disks with lots of holes and move its used RBs to other disks. Resulting empty PEG is now usable by RAID5.
- What if there aren’t enough holes? Write the excess RBs to RAID5, then reclaim the PEG.
- Migration: which RBs to demote? Least-recently-**written (not LRU)**
- Balancing: make sure data evenly spread across the disks. (Most important when you add a new disk)

Bad cases? One is thrashing when the working set is bigger than the mirrored storage

Performance:

- consistently better than regular RAID, comparable to plain disks (but worse)
- They couldn’t get traditional RAIDs to work well...

Other things:

- “shortest seek” -- pick the disk (of 2) whose head is closet to the block
- When idle, plugs holes in RAID5 rather than append to log (easier because all RBs are the same size!) Why not all the time? Requires reading the rest of the stripe and recalculating parity
- Very important that the behavior is dynamic: makes it robust across workloads, across technology changes, and across the addition of new disks. Greatly simplifies management of the disk system

Key features of paper:

- RAIDs difficult to use well -- two levels and automatic data movement simplifies it
- Mix mirroring and RAID5 automatically
- Hide magic behind simple SCSI interface