RAID

Originated as organizational techniques for disks, but now more general... How can we organize devices and what extra state should we keep to increase both reliability and performance?

Reliability via redundancy

MTTF (mean time to failure)... 1 disk = 100,000 hours MTTF

100 disks = 100,000/100 = 1,000 hours = 41.66 days MTTF of *some* disk

Bad. **Mirror** every disk... **MTTR** (mean time to recovery), say 10 hours

assuming disk failures are independent events then **mean time to data loss** is...

$(100,000/10)^2$ or about 57,000 years. Not bad, but

OK to assume independent failures? No, but still reliable.

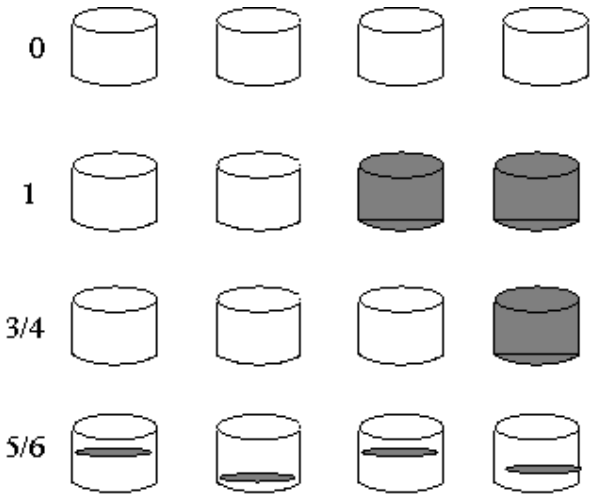
Performance via parallelism

Does just mirroring improve read throughput? 2X *Write?* No

How can we improve write throughput?

Stripe data across multiple disks.

- Bit level − reduce the response time of large accesses
- Block level − increase throughput of multiple small accesses by load balancing
- time/block vs. IO operations/sec



RAID 0 Striping, no redundancy

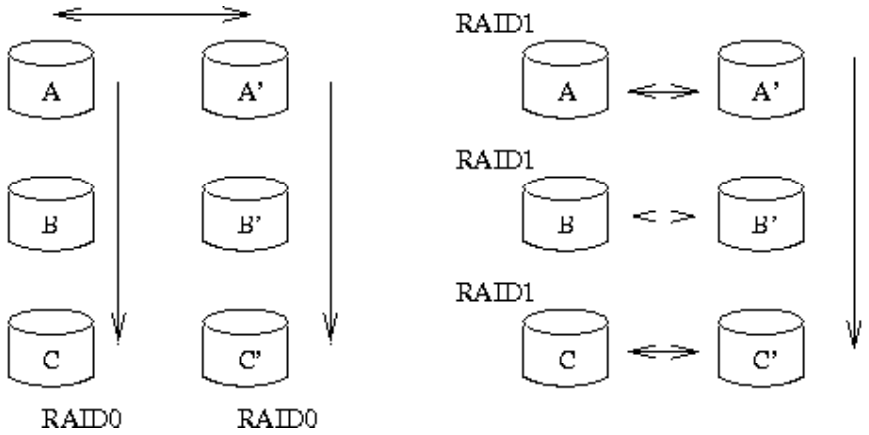
RAID 1 Mirroring

RAID 0+1

- Mirrored pair of two striped sets
- Fails if one disk in each set fails

RAID 1+0

- A striped set of N mirrored pairs
- Fails if both disks in one set fail



Which is better?

RAID 2 memory−like ECC

- Multiple parity disks

RAID 3 bit interleaved parity

- Unlike memory, disk controllers can detect which sector read incorrectly
- ...so only need 1 parity disk
- Every disk participates in every I/O request
- Fast block read/write, but fewer independent I/O ops per second
- Calculating parity usually done in HW

RAID 4 − block level striping and parity disk

- Small independent writes cannot be performed in parallel
- Because contention for parity disk
- Single write requires 4 disk accesses (2 read old, 2 write new)
- Easy to add disk, just set it to all zeros. Example: WAFL
- Problem: overruse of parity disk

RAID 5 − block interleaved disk parity

- Spread parity among all disks
- Avoid overuse of parity disks
- Can not store parity for block in same disk. *Why?*

RAID 6 − P + Q

- Store extra redundant information to tolerate multiple disk failures
- Error correcting codes, e.g. Reed−Solomon

Commonly in use

- Commodity motherboards support levels 0 and 1
- Level 0+1, 1+0 for performance and reliablity
- Level 5 if space overhead an issue (large storage)
- RAID−like techniques built into FS (GFS)

Important issues

- Rebuild time
- Hot spares
- Replication (to other sites: trust issues, natural disasters)

Dealing with corruption

- Software errors − kernel and FS bugs
- Hardware errors − no error but returns corrupted data
- Internal checksums of all blocks, inodes, etc.
- Example: ZFS. Checksum for block B stored with inode that points to B