Block-level RAID is dead

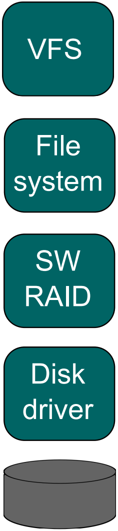
Raja Appuswamy, David C. van Moolenbroek,
Andrew S. Tanenbaum

Vrije Universiteit, Amsterdam

June 22, 2010

Traditional storage stack

- Originally one file system per disk
- Later RAID layer was introduced
  - Block-level RAID and Volume managers
- Storage stack has remained the same for decades
- Compatibility-driven integration has fatal flaws



VFS

File system

SW RAID

Disk driver

Problem 1: Silent data corruption

- Disks exhibit fail-partial failure modes
  - Lost, torn, misdirected writes
  - Such failures result in silent data corruption
- Checksumming algorithms fail to detect corruption
  - Most algorithms detect only a subset of all failure modes
  - Parental checksumming detects all classes of failures
- Parental checksumming fails with block-level RAID
  - RAID-initiated reads are unverified
  - RAID-initiated reads propagate corruption

Problem 2: Heterogeneity issues

- Integration of new devices is an interesting problem

- Building device-specific FS
  - Not compatible with block-based RAID

- Building a translation layer
  - Widens the “Information gap”
  - Duplication of functionality

Problem 3: Device failure

- Traditional RAID fails ungracefully
- Graceful degradation has two requirements
  - Selective metadata replication
  - Fault-isolated file placement
- Semantically unaware traditional RAID cannot fail gracefully

Problem 4: Administration nightmare

- Too many Volume management abstractions
  - PVs, VGs, LVs, FSes, etc.
  - Simple tasks need several error-prone steps
- Too many tunable parameters
  - Chunk size, stripe width, LV size, etc.
  - Improper configuration leads to bad performance
- Coarse-grained policy specification
  - Need more flexibility (per file, directory or volume)

Problem 5: System failure

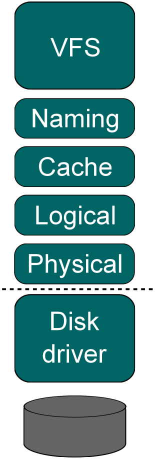
- Crashes/power failures result in “Write holes”
- HW RAID uses NVRAM to sidestep this issue
- Software RAID cannot rely on NVRAM
  - Whole-disk resynchronization is impractical
  - Journaling duplicates functionality and affects performance

Loris - the new storage stack

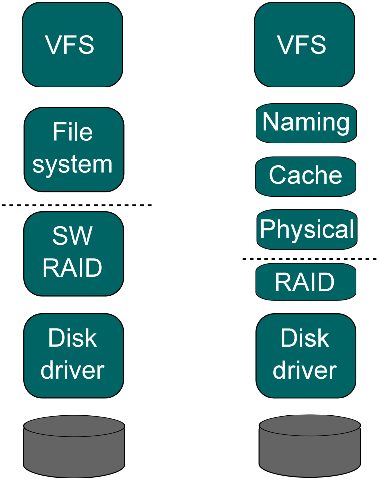
- File-based interface between layers
    - Each file has a unique file identifier
    - Each file has a set of attributes

- File-oriented requests:

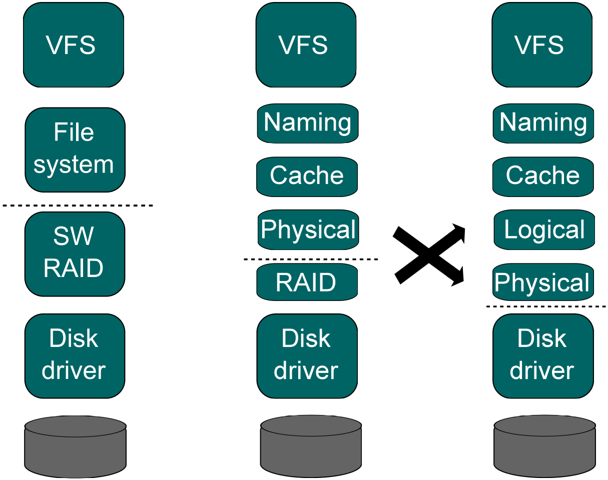
    create
    delete
    read
    write
    truncate
    getattr
    setattr
    sync



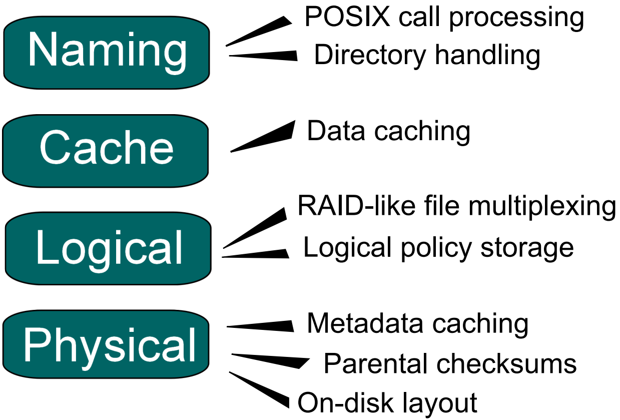
Modular split and reliable flip (1)



Modular split and reliable flip (2)



Loris - the new storage stack



Naming

- POSIX call processing
- Directory handling

Cache

- Data caching

Logical

- RAID-like file multiplexing
- Logical policy storage

Physical

- Metadata caching
- Parental checksums
- On-disk layout

Solution to problem 1: End-to-end data integrity

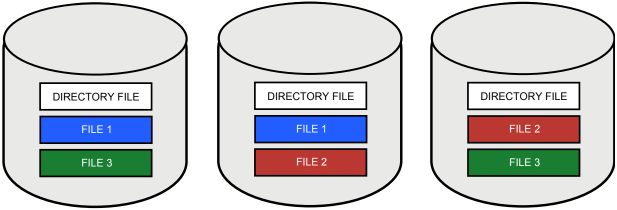
- Physical layer converts fail-partial to fail-stop failures
- Physical layer verifies all requests alike
- RAID algorithms provide recovery from fail-stop failures

Solution to problem 2: Embracing heterogeneity

- Device-specific physical layers
  - Can exploit device access characteristics
  - Eliminate multiple translation steps
- RAID and Volume management across device families
  - File abstraction hides device-specific vagaries
  - No need to reimplement RAID algorithms per device family

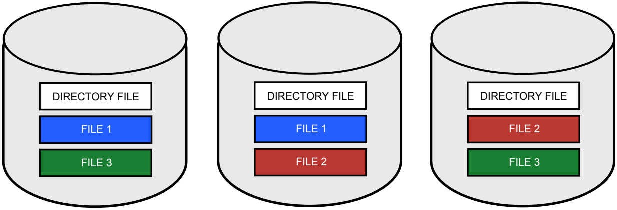
Solution to problem 3: Graceful failure

- Directories replicated on all devices
  - Naming layer chooses RAID 1 policy
- Zero-effort fault-isolated placement



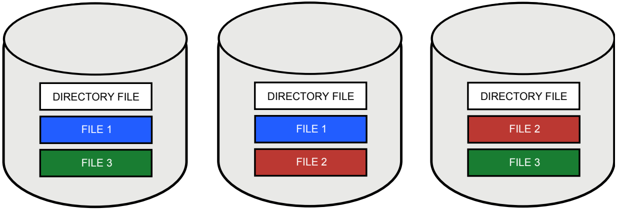
Solution to problem 3: Graceful failure

- Directories replicated on all devices
  - Naming layer chooses RAID 1 policy
- Zero-effort fault-isolated placement



Solution to problem 3: Graceful failure

- Directories replicated on all devices
  - Naming layer chooses RAID 1 policy
- Zero-effort fault-isolated placement



66% availability under two failures!

Solution to problem 4: Simplified administration

- File pools similar to storage pools
  - New device ⇒ new source of files
  - Completely automate error-prone tasks
  - “File systems/Volumes” share the file pool
- Flexible policy assignment
  - Logical layer provides mechanism
  - Any layer can assign policies
  - Policies per file, directory, or volume

Solution to problem 5: Crash recovery

- Traditional FS recovery techniques can be used
  - Journaling in physical layer (ext3)
  - Transactional COW (ZFS)
- Goal is to protect important user data
  - Metadata journaling does not help
  - Full data journaling is very expensive
  - Can we do selective data journaling?

Conclusion

- We examined block-level RAID along several dimensions
- We highlighted several fatal flaws
- We suggested a simple, yet fundamental change to the stack
- We showed how the new stack solves all issues by design