Challenges of Distributed Storage with Data Protection

Director of Algorithms, Excelero

Daniel Herman Shmulyan
Who is Daniel H. Sh.?

Director of Algorithms at Excelero

- 15 years of software development and architecture experience in various aspects of high performance, multi-platform and distributed systems including machine learning, storage and computer vision.
- Links: [Linkedin](#)  [Stackoverflow](#)
- [www.excelero.com](#)
Distributed Shared Storage HW Setup

- Single rack
- Clients: Hosts/Initiators (CPU, NIC, RAM)
- Targets: Servers (+NVME Drives)
  - NVRAM, NV-DIMMs, Cache
- RDMA notation
- RAID6 (say 8+2)
Full dis-aggregation

 Targets: hold full state
 Clients: do the full compute

- Target CPU is idle in data path
- Stateless client
  - $C_1$ crashes, $C_2$ continues
Benefits of dis-aggregation

- Scalability: Storage / IOPs / Recoveries
  - 50K clients <---> 500 Targets
- Mathematical fairness
- Easy migration
- Control path is not a bottleneck to datapath
Data Path Basics

- **Transactionality of 4K I/O:**
  - Client managed journaling
  - Mutually exclusive clients

- **State:**
  - Drive Block & Metadata
  - Volatile RAM (Locks, etc)
    - Same redundancy as data on disks

**Execution Plans**

- **Basic I/O** (good path, degraded mode)
- **Failed I/O fixups** (client disconnection fixups)
- **Raid-Rebuilds** (boot, recoveries)
- **Maintenance** (scrubbing, wrap-around, migration)
- **Control Path**
Components of EC algorithm

- Virtual Block mapping
  - May differently for 2 cpu cores
- Cmds to drives
- Locks + RAM data structures
  - 1+P backup
- Journals
  - Write atomicity, no write-hole
  - IO atomicity
  - Written to same disks as IO
## Execution Plan for data-path

### Write of 1[blk] execution

1. Build plan {VLBA, topology}
2. Acquire locks
3. Reads old data
4. Locally allocate journals
5. RAID6 GF/CRC calculation
6. Write journals to drives
7. Update RAM data structures
   --- Here RollFwd is possible ---
8. Commit Data to drives
9. Unlock the lock

### Failed I/O fixup By C₁

1. C₁ (recoverer), C₂ (recoveree)
2. C₁ detects abandoned lock of C₂
3. Acquire locks
4. Get const access to C₂ resources
5. Analyzes RAM, journals, drives
6. Solve write-holes hazards
7. Solve Topology C₁-C₂ discrepancy
8. Solve natural disasters
9. Solve Topology transitions

### Issues: Simplicity, Performance
Raid 4+2 Example

- $C_1$ 8K Write: $\{D_0, D_1, P, Q\}$
- $C_2$ “roll-fwds” $D_0$
- $C_2$ updates dirty markers
- Crash in “roll-fwd/bkwrdo”
- Cold Recovery?

$\{D_0, P\}$ succeeds
Summary of the Challenges

- No \{C_1, C_2\} Talks
- Control path topo barriers, with datapath.
- Scalability: Distributed Data and Control Path
- u64 rdma cmp-xchng – fast & atomic
- Datapath does not wipe journals
- Locks Topologies
- Single block IO write amplification
Raid 8+2 Topologies

- Drive has 3+ topos {Dead, Rebuild, OK}
- Total Raid topologies: 201
- Clients \(\{C_1, C_2\}\) Topologies matrix 201\(^2\).
  - Contradicting Topologies
  - Control Path: soft barrier on Topologies
- Locks Topologies

\[4C^P_{D+P} + 2(D+P)+1\]