Erasure Coding & Read/Write Separation in Flash Storage

Dimitris Skourtis, Dimitris Achlioptas, Noah Watkins, Carlos Maltzahn, Scott Brandt
University of California, Santa Cruz

INFLOW'14
Outline

• Background
  • Performance (un)predictability in SSDs
  • Making SSD storage predictable with Rails

• Erasure Coding and Rails
  • Coding performance
  • Throughput and scaling

• Evaluation

• Conclusion
SSD Performance
Unpredictability
Reads are perfect
Writes are unpredictable
Physical read/write separation with Rails
Basic 2 drive design

- Reads
- Writes

Cache

- Writes of current period
- Writes from previous period

SSD 1
Read mode

SSD 2
Write mode

Drives stay in sync (periodically)
Design properties

- Consistency
- Heterogeneity
- Power failure
- Twice the cost?

Cache + SSD 1 = Cache + SSD 2
Design generalization

![Diagram showing frame i and frame i+1 with readers and writers]

Large storage arrays and distributed storage already employ redundancy (replication or erasure coding)
An object is spread among nodes depending on the redundancy.

Object O (write)

Controller

RAM

Flash

Node 1 …

Readers window

Node 6

Reading nodes accumulate writes in RAM.
Flushing writes

Write nodes flush data to drives

$D_{i-4,i-3,i-2}^{(1)}$ flushed during frame $i$

$D_{i-4...i}^{(1)}$ to $D_i^{(6)}$
Rails & Erasure Coding
Rails & erasure coding

- Avoid storage space overhead of replication
- Perform reads through reconstruction (decoding)
  - Utilizes current set of drives dedicated to reading
  - Pay in computational cost
- Scale by constructing redundancy groups
  - Computational cost scalable
  - Maintain read/write separation
Erasure coding

- Write object of size 100MB
- Obfuscate (encode) to 120MB
- Split into 12 chunks of 10MB each
- Distribute across 12 drives
- Any 10 drives/chunks may be used to read the original object
Decoding throughput

Decoding throughput \([k = m, k \text{ chunks unavailable}]\)
Throughput in #threads

Decoding throughput [ $k = m$, $k$ chunks unavailable ]
Read throughput without/with decoding [k=m, k drives unavailable]
Redundancy groups with R/W separation

Hypergraph with four overlapping hyperedges (redundancy groups), each containing three vertices (drives)
Generating redundancy groups for R/W separation

\[ P_i = \text{drives of color } C_i \]
Write throughput bound
Performance of eRails (erasure coding, 6 drives)

Without eRails  
With eRails
Performance of eRails (erasure coding, 10 drives)

Without eRails

With eRails
Summary

• Erasure coding
  • Space-efficient redundancy method for Rails
• Computational cost
  • Increase to the array size
  • After certain #drives (e.g., more than 10) throughput decreases
• Achieve scaling
  • Generate overlapping redundancy groups
  • Leads to proportional increase of computational cost