Rethinking Erasure Codes for Cloud File Systems: Minimizing I/O for Recovery and Degraded Reads
Osama Khan and Randal Burns, Johns Hopkins University, James Plank and William Pierce, University of Tennessee
Cheng Huang, Microsoft Research

Growing storage demands make replication too expensive
The data explosion phenomenon has led people to consider using erasure coding in place of replication. Erasure coding offers similar fault tolerance as replication but at a much lower storage cost.

Erasure Coded Storage Systems
Cloud file systems use large block sizes. When full, each block is sealed, erasure coded, and distributed to storage nodes. Data is encoded in units of stripes, using a generator matrix, and is parameterized by k, m, and r. Within a stripe, data is broken up into symbols.

Algorithm to minimize recovery I/O

- A decoding equation is a set of symbols whose corresponding rows in the matrix sum to zero.
- Enumerate all valid decoding equations for each failed symbol
- Construct a directed graph where:
  - nodes are bit strings
  - edges denote equations
  - child’s bit string is bitwise OR of parent’s bit string and equation on incoming edge
  - Shortest path through graph gives set of equations which would minimize recovery I/O
- Traversing a level is equivalent to recovering a failed symbol

Example

- Suppose R_i and R_j fail.
- Enumerate the decoding equations for each symbol
- Construct the graph on the right.
  - The edges along the shortest path are highlighted in bold (3 + 2 = 5 symbols)
  - May have more than one shortest path

Disk Reconstruction and Degraded Reads
Device failures are common at such large scales, so data recovery is frequently needed. Two operations emerge out of this need.

- Disk reconstruction: failed disk is reconstructed in its entirety
- Degraded read: read request has a failed disk within its span, so retrieves missing data using the erasure code
It is to be noted that nodes can go down not just due to device failures, but also due to rolling software updates.

What is the problem?
Existing erasure codes were not designed with recovery I/O optimization in mind. So we need:
- To optimize existing codes for these operations
- New codes which are intrinsically designed to optimize these operations

Rotated Reed Solomon Codes

- Derived from standard Reed-Solomon codes.
- Optimized for recovery from single disk failures
- Performance compared against standard Reed-Solomon Codes, which use matrix inversion to recover from failures (equivalent to reading from the parity drive P, in terms of the number of symbols read)

Degraded Read Example

- Read request of 4 symbols starting at d_{10}
- Penalty = 8 of symbols read - read size

Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>Disk Reconstruction cost</th>
<th>Degraded Read penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using Rotated Reed</td>
<td>16 symbols read</td>
<td>2 symbols</td>
</tr>
<tr>
<td>Solomon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Using P Drive</td>
<td>24 symbols read</td>
<td>5 symbols</td>
</tr>
</tbody>
</table>

Conclusions

- Generally, optimally sparse and minimum density codes perform best
- Rotated Reed-Solomon codes are a better alternative to standard Reed-Solomon codes for cloud storage
- Traditional RAID configurations (small sized blocks) do not yield good recovery performance with cloud based storage systems due to seek penalty
- Our algorithm is effective only for large symbols. Although HDFs and others already use a default size of 64MB, even larger sized sealed blocks are recommended (at least 100 MB, preferably > 500MB)
- Minimizing the number of symbols needed for recovery does result in lower I/O cost