Rethinking Erasure Codes for Cloud File Systems: Minimizing I/O for Recovery and Degraded Reads

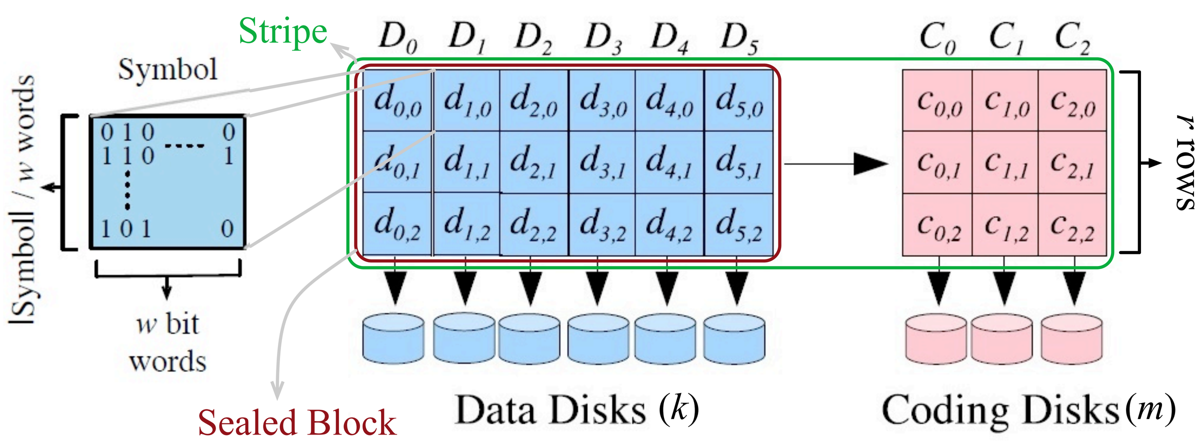
Osama Khan and Randal Burns, Johns Hopkins University, *James Plank and William Pierce,* University of Tennessee
Cheng Huang, Microsoft Research

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*.



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.

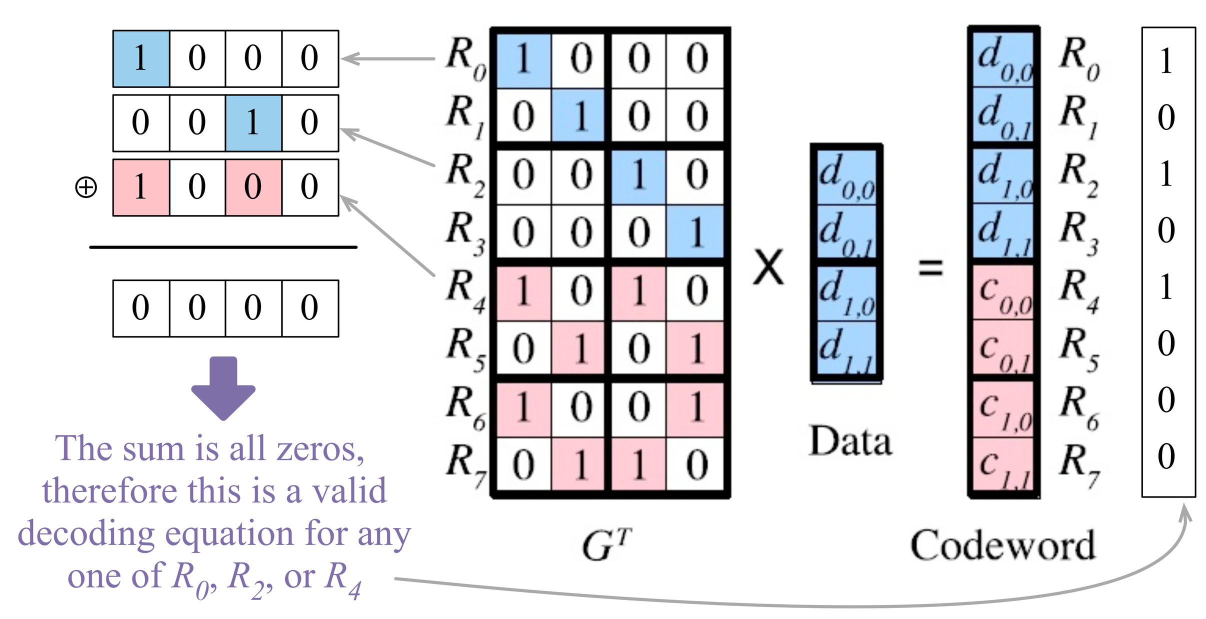
What is the problem?

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

Algorithm to minimize recovery I/O

- A decoding equation is a set of symbols whose corresponding rows in the matrix sum to zero.



Works for *any* XOR based code!!

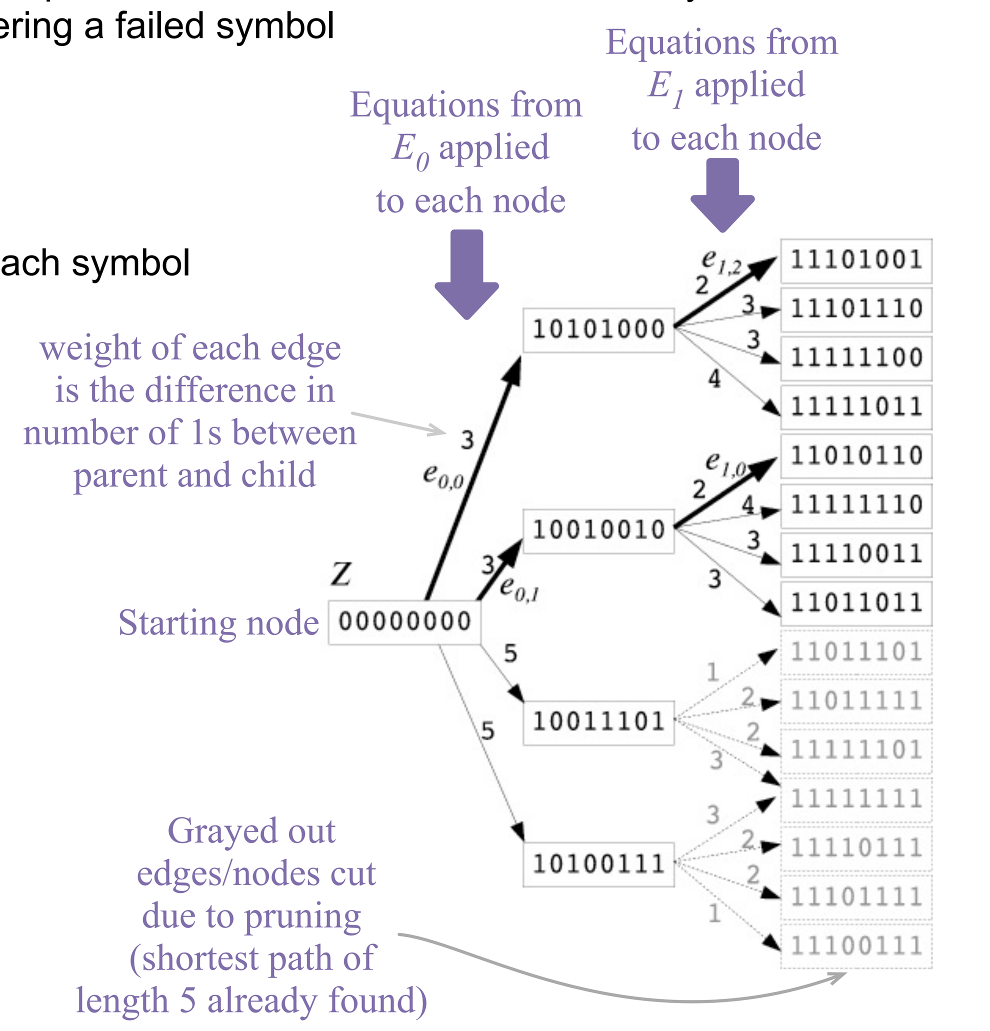
- Enumerate all valid decoding equations for each failed symbol
- Construct a directed graph where:
  - nodes are bit strings
  - edges denote equations
  - child's bit string = bitwise OR of parent's bit string and equation on incoming edge
- Shortest path through graph gives set of equations which would minimize recovery I/O
- Traversing a level is equivalent to recovering a failed symbol

Example

- Suppose R_0 and R_1 fail.
- Enumerate the decoding equations for each symbol

E_0	E_1
$e_{0,0}$ = 10101000	$e_{1,0}$ = 01010100
$e_{0,1}$ = 10010010	$e_{1,1}$ = 01101110
$e_{0,2}$ = 10011101	$e_{1,2}$ = 01100001
$e_{0,3}$ = 10100111	$e_{1,3}$ = 01011011

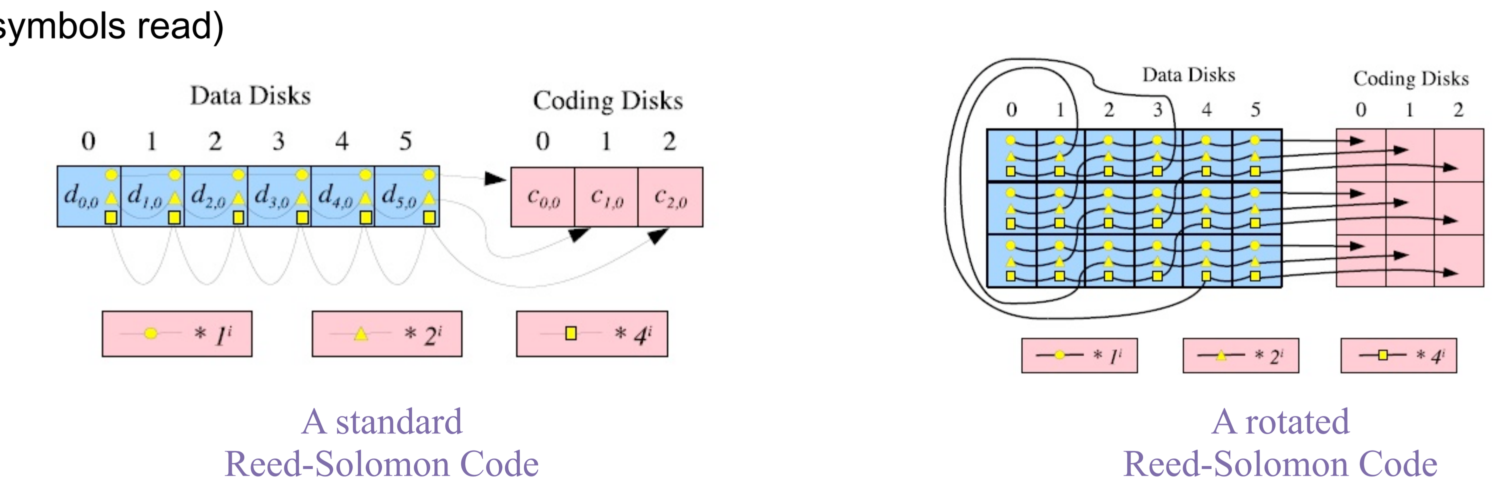
Recovery options for R_0 — Recovery options for R_1



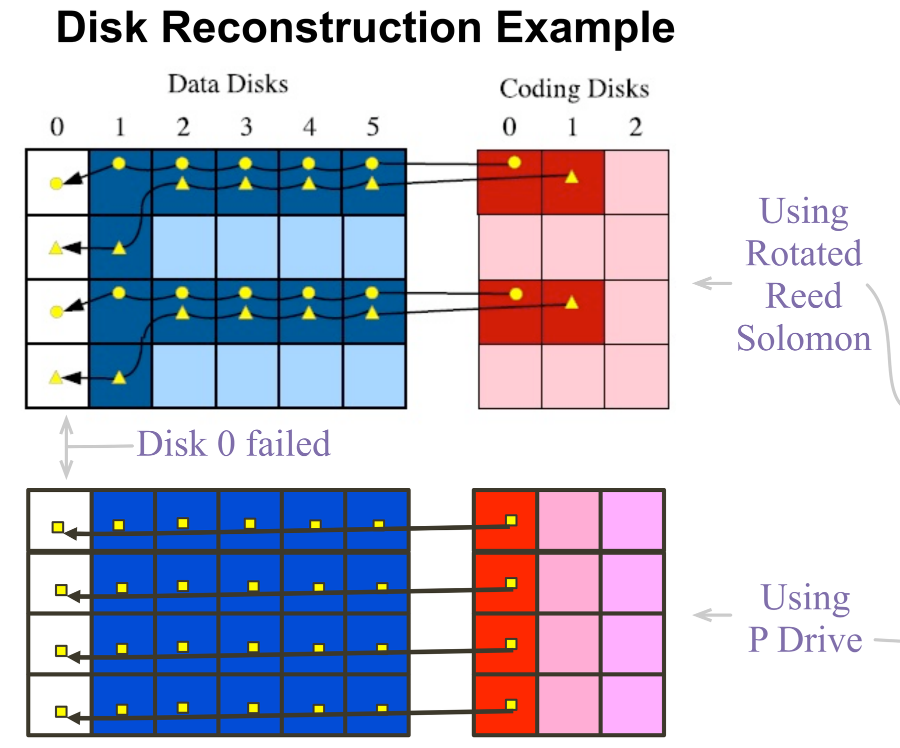
- Construct the graph on the right
- The edges along the shortest path are highlighted in bold (3 + 2 = 5 symbols)
- May have more than one shortest path

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)

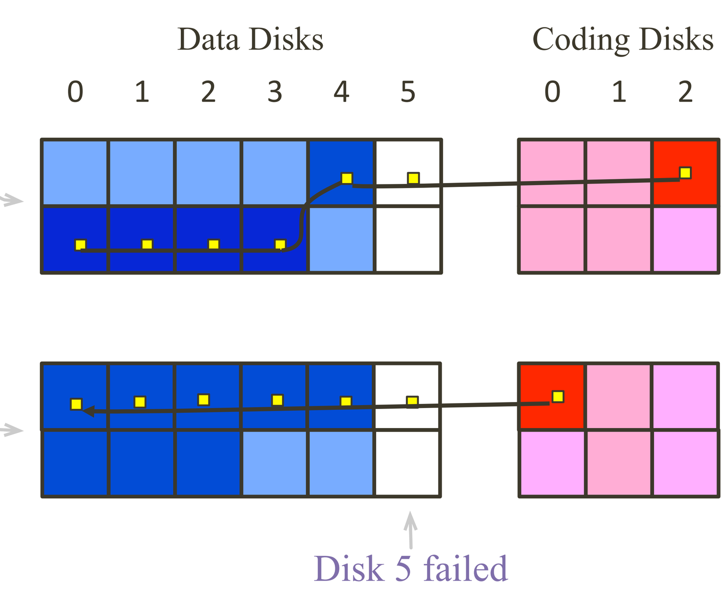


Disk Reconstruction Example



Degraded Read Example

- Read request of 4 symbols starting at $d_{5,0}$
- Penalty = # of symbols read - read size



Coding symbol not read — Data symbol not read — Coding symbol read — Data symbol read

Performance Comparison

	Disk Reconstruction cost	Degraded Read penalty
Using Rotated Reed Solomon	16 symbols read	2 symbols
Using P Drive	24 symbols read	5 symbols

Analysis

Disk Reconstruction

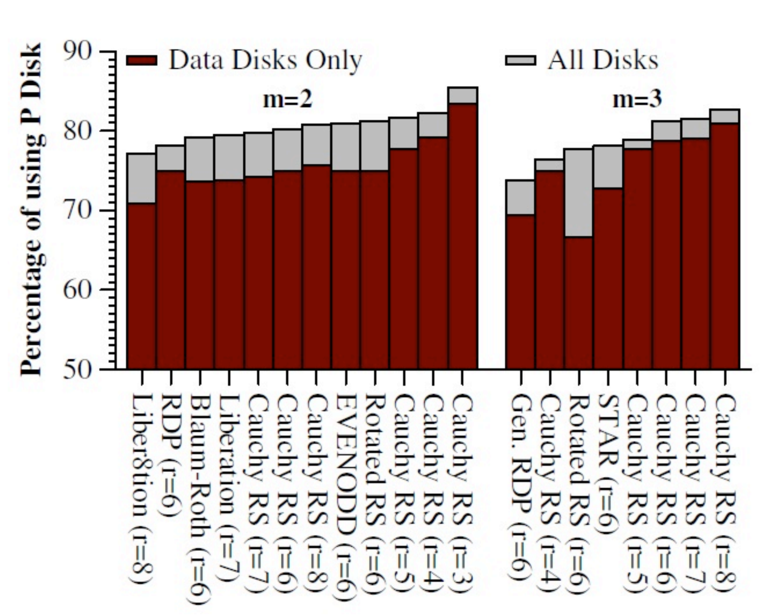


Figure: Number of symbols read during recovery using our algorithm as a percentage of standard recovery

- Best reconstruction performance given by Liber8tion codes (*m* = 2), and Generalized RDP (*m* = 3)
- Standard (Cauchy) Reed-Solomon codes have high recovery cost in cloud storage systems

Degraded Reads

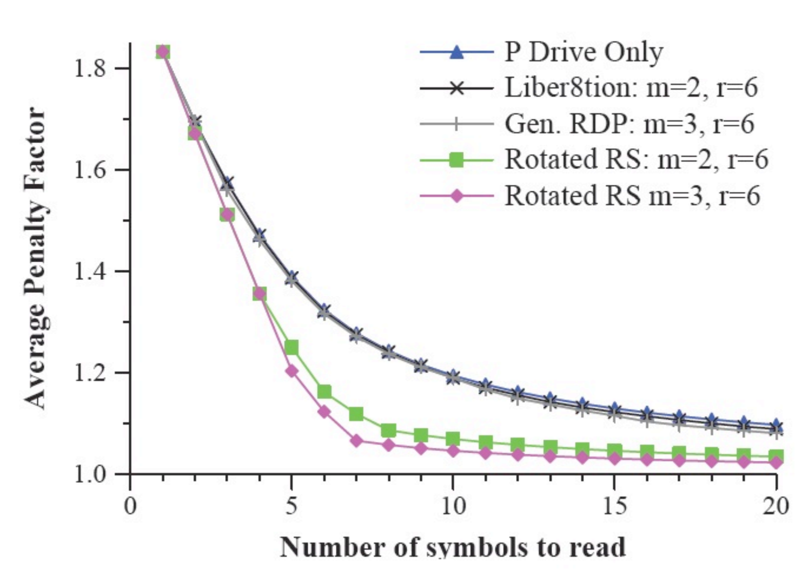


Figure: Average over all (*k* = 6) data disks failing and over all *kr* potential starting points in the stripe

- Rotated Reed-Solomon codes better than optimally sparse and minimum density codes
- Recovery for single symbol requests requires all codes to read *k* symbols.
- Degraded reads of entire stripes incur no penalty as read request already contains symbols needed for recovery

Sealed Block Size

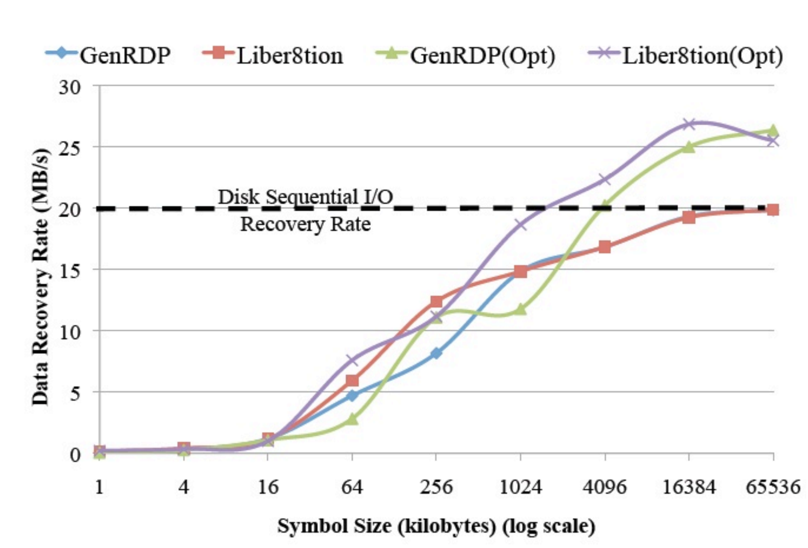


Figure: Recovery performance measured as a function of varying symbol sizes (and indirectly, sealed block sizes)

- Traditional recovery performance of Generalized RDP and Liber8tion codes is compared with the optimized versions
- At larger symbol sizes (> 4 MB), recovery with the optimized version is faster than the P drive based streaming recovery rate

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 sealed 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