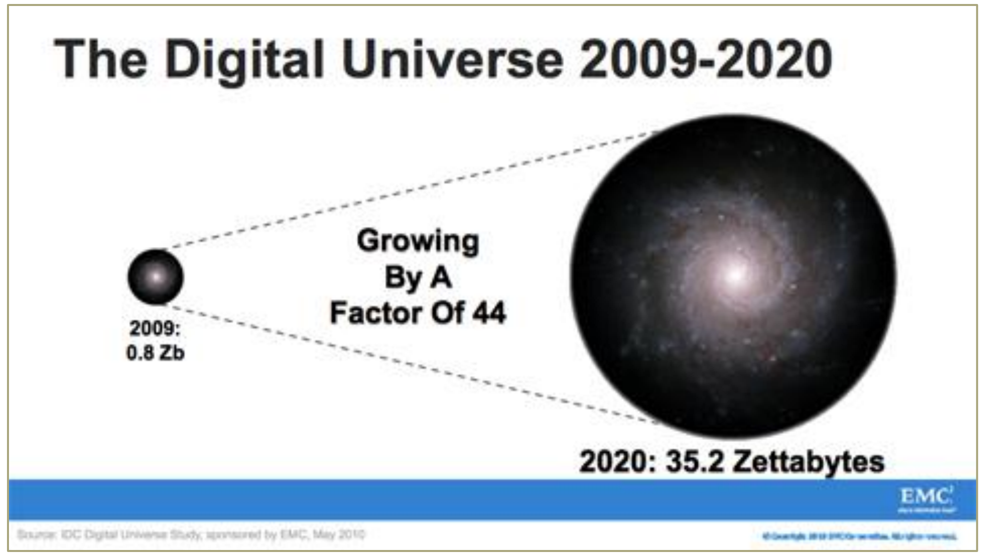
Rethinking Erasure Codes for Cloud File Systems: Minimizing I/O for Recovery and Degraded Reads

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

USENIX FAST 2012

What is the problem?

- Data Explosion



- Much of that data will be stored in the cloud
- Replication too expensive $\rightarrow$ Erasure coding to the rescue
  - As pointed out previously [Zhang '10 and others]

What is the problem?

- Humongous scale + failure rates = Frequent recovery needed
  - Also, rolling software updates result in downtime [Brewer '01]
- Two operations become prominent:
  - **Disk reconstruction**
  - **Degraded reads**
- Existing erasure codes are not designed with recovery I/O optimization in mind
  - Need to optimize existing codes for these operations
  - Need new codes which are intrinsically designed for these operations

Minimizing Recovery I/O

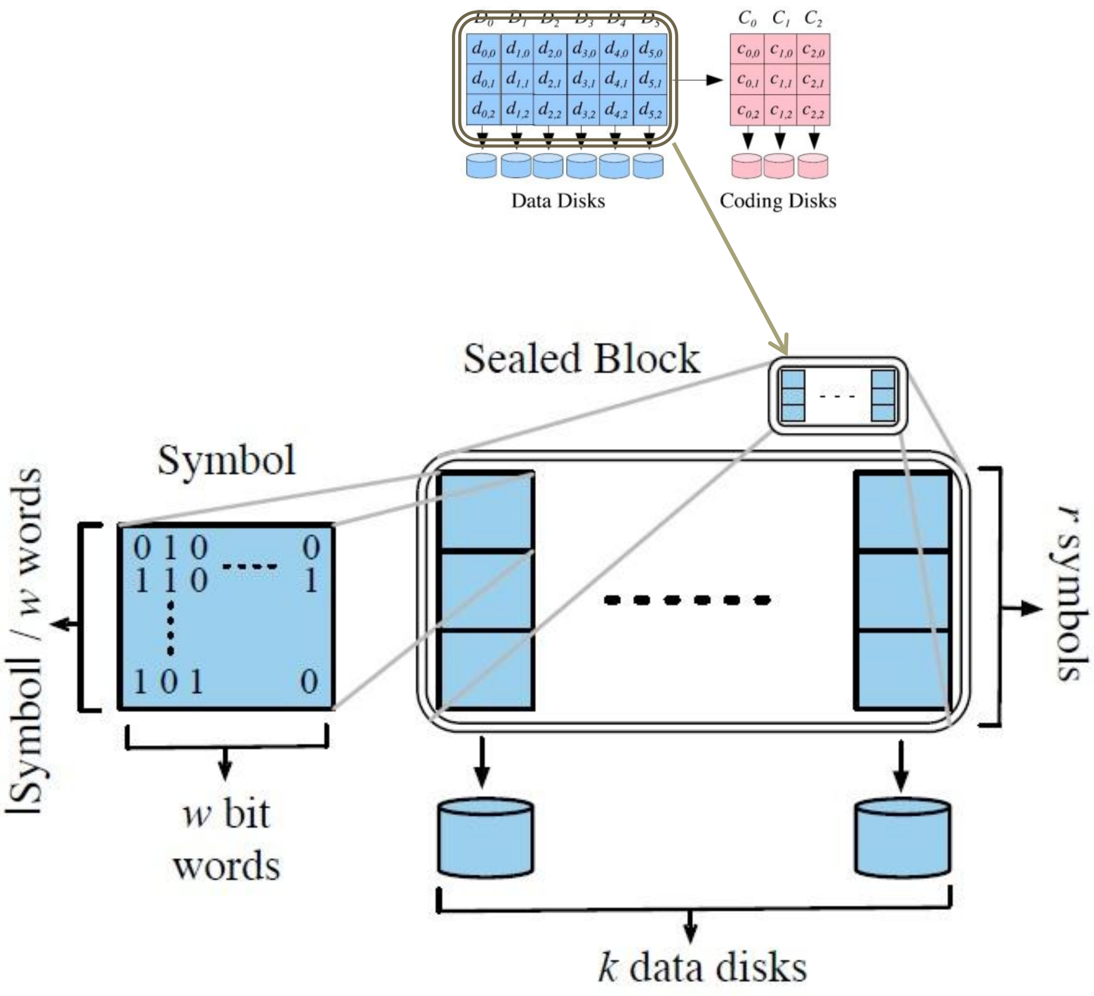
- Algorithm minimizes the amount of data needed for recovery
  - Applicable to **any** XOR based erasure code
- Existing erasure codes and configurations are not suitable for the cloud
  - Large file system blocks required to extract good recovery performance
- Rotated Reed-Solomon Codes
  - A new class of Reed-Solomon Codes which optimize degraded read performance
  - Better choice than standard Reed-Solomon codes for the cloud

Outline

- Erasure Coded Storage Systems
- Algorithm for minimizing number of symbols
- Rotated Reed-Solomon Codes
- Analysis & Evaluation
- Conclusions

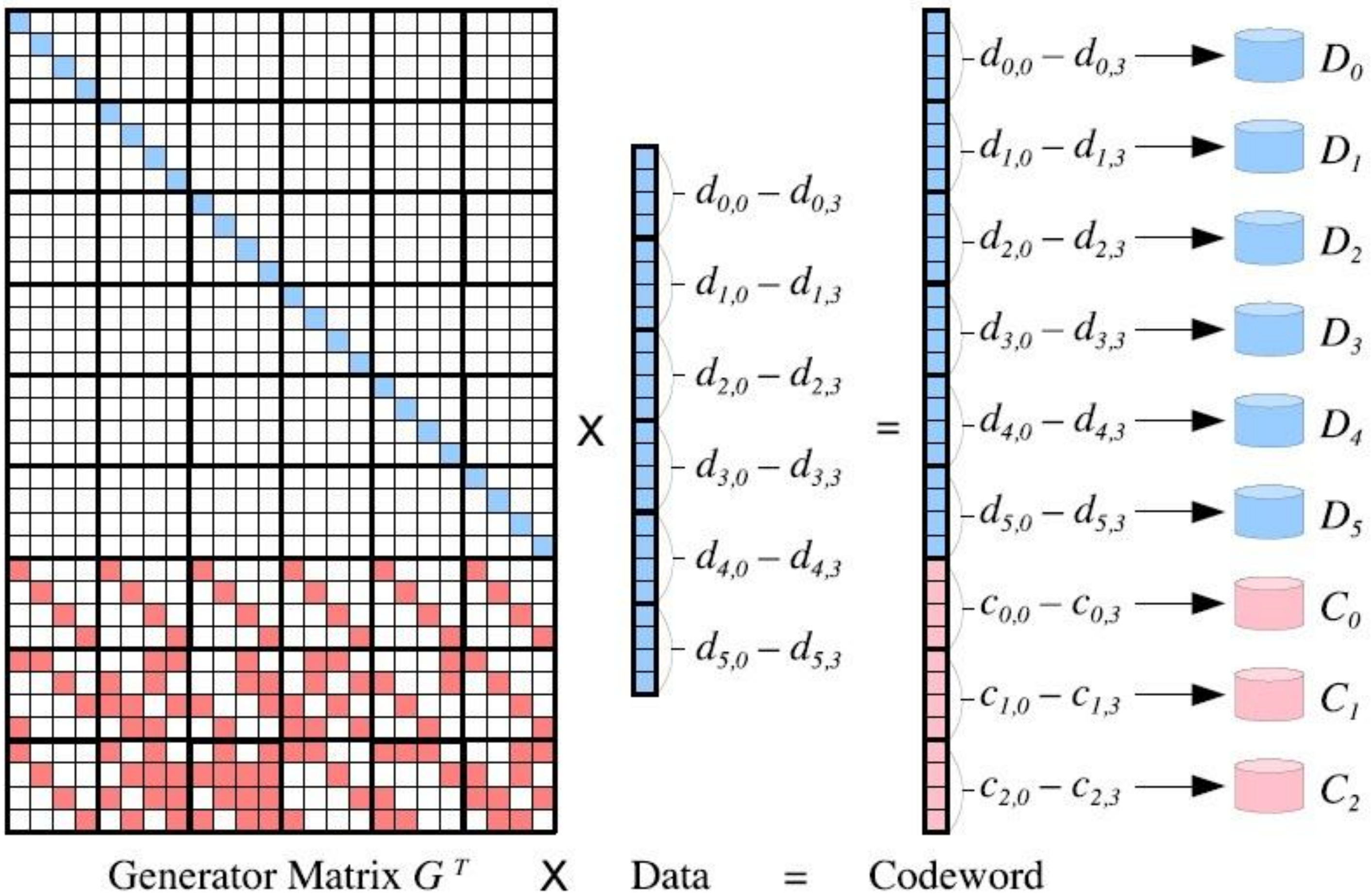
Erasure Coded Storage Systems

Wait until block is full → Sealed → Erasure coded → Distributed to nodes



Erasure Coded Storage Systems

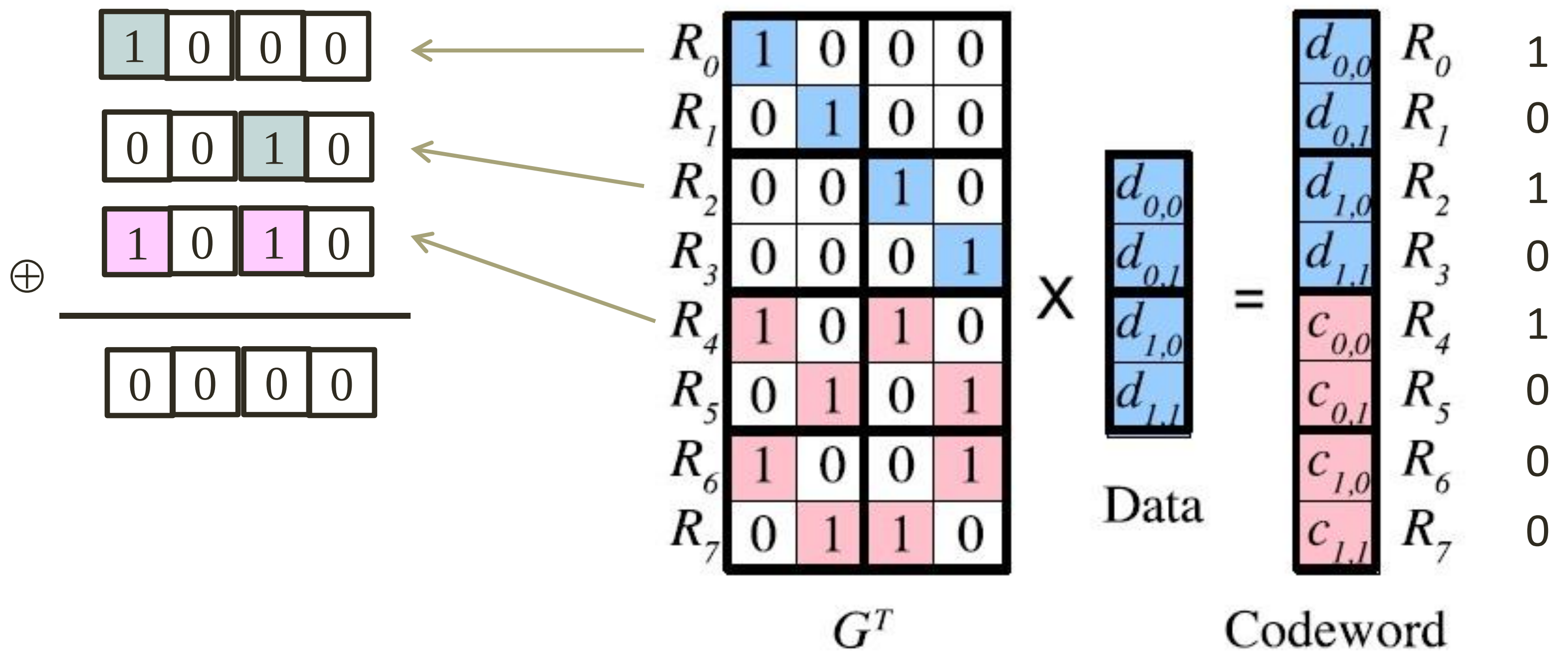
k = 6 m = 3 r = 4



Outline

- Erasure Coded Storage Systems
- **Algorithm for minimizing number of symbols**
- Rotated Reed-Solomon Codes
- Analysis & Evaluation
- Conclusions

Decoding Equations



{R_0, R_2, R_4} is a **decoding equation**

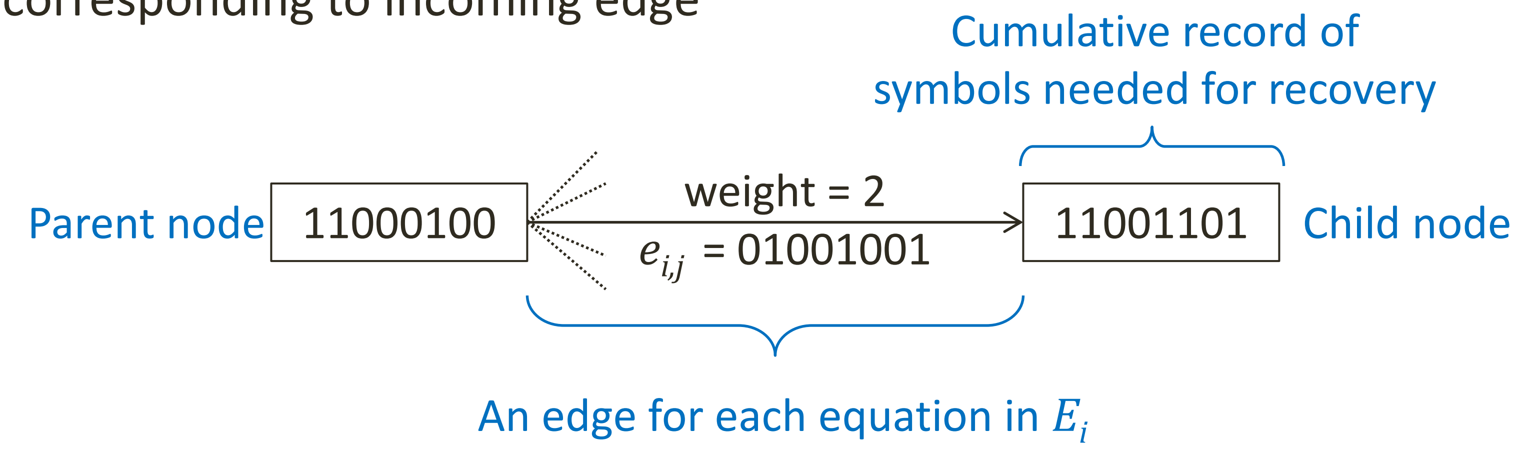
And it can be represented by 10101000

Algorithm to minimize recovery I/O

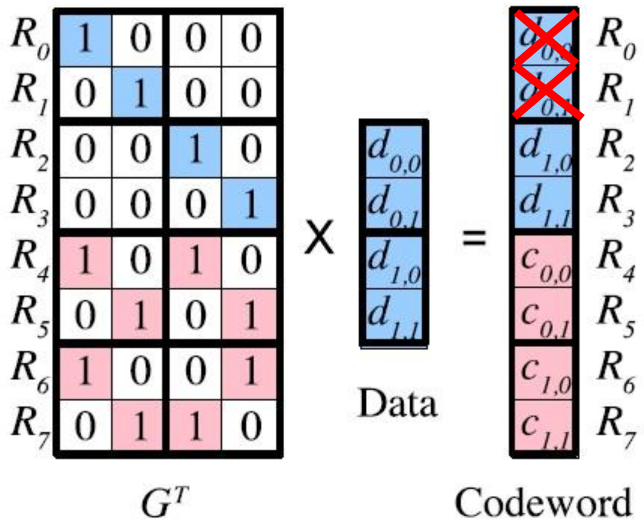
- Finds a decoding equation for each failed bit while minimizing the number of total symbols accessed
- Makes use of data sharing [Xiang ‘10]
- Given a code generator matrix and a list of failed symbols, the algorithm outputs decoding equations to recover each failed symbol

Algorithm Details

- Enumerate all valid decoding equations for each failed symbol
- Directed graph formulation of problem makes it convenient to solve
  - Nodes are bit strings
  - Edges denote equations
  - Child's bit string = parent's bit string OR'ed with equation corresponding to incoming edge



Example



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

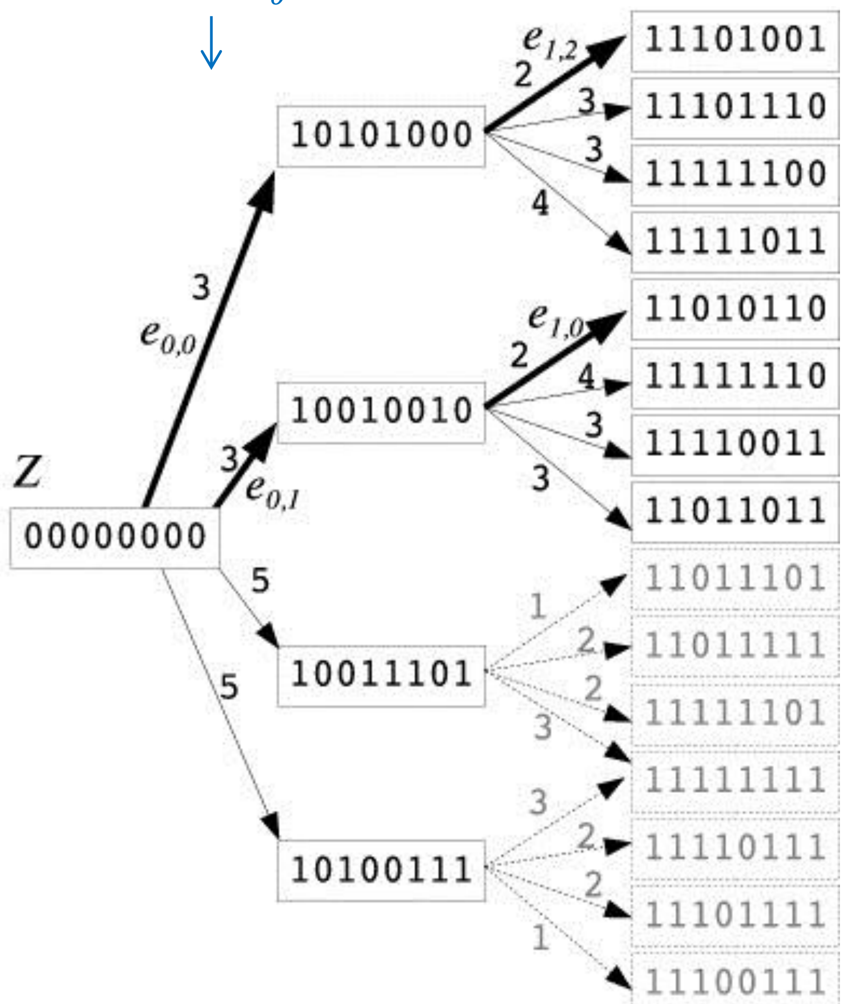
Recovery options for R_1

Example - Graph

Level 0: Equations from E_0

Level 1: Equations from E_1

Starting node



Grayed out nodes/edges denote pruning

Algorithm Summary

- Minimizes the number of symbols needed to recover from an arbitrary number of failures
- Solutions to all common failure combinations may be computed offline *a priori* and stored for future use
- Works for any XOR-based code
  - Generalizes previous results (EVEN/ODD[Wang '10], RDP[Xiang '10])
  - Other codes turned out to perform better than EVEN/ODD and RDP

Outline

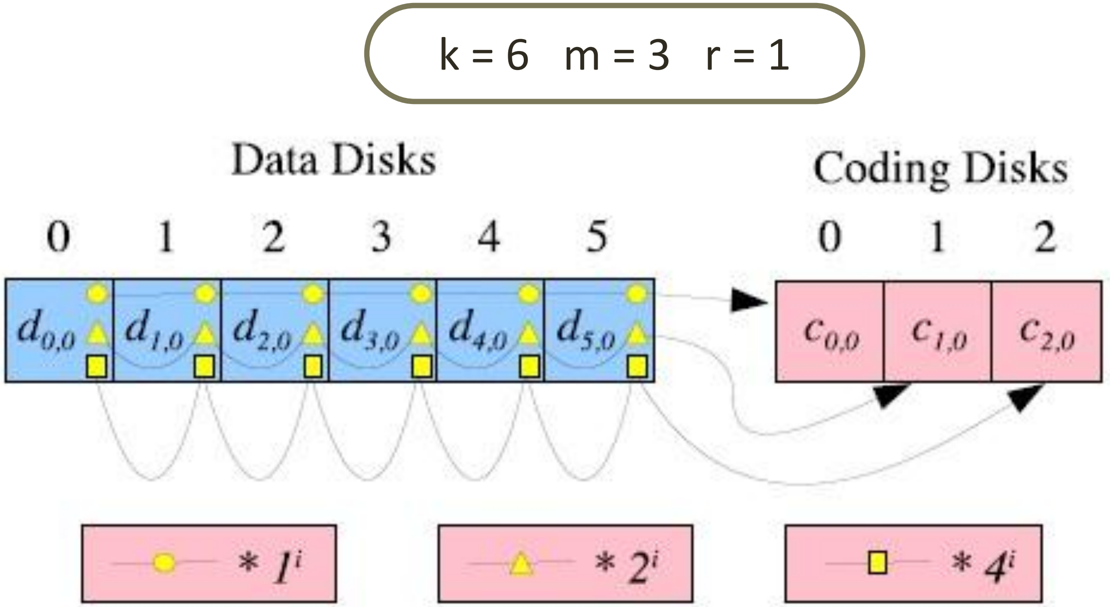
- Erasure Coded Storage Systems
- Algorithm for minimizing number of symbols
- **Rotated Reed-Solomon Codes**
- Analysis & Evaluation
- Conclusions

Rotated Reed-Solomon Codes

- Vast majority of failure scenarios are single disk failures (99.75% [Schroeder '07])
- 90% of failures are transient and do not involve data loss [Ford '10]
  - Google waits 15 minutes before reconstructing disk
  - Degraded read to missing data requires recovery using erasure code
- New class of codes optimize degraded read performance in case of single disk failure
  - MDS (for certain values of k, m and r)
  - Modification to standard Reed-Solomon codes

Standard Reed-Solomon Codes

- A sample Reed-Solomon code



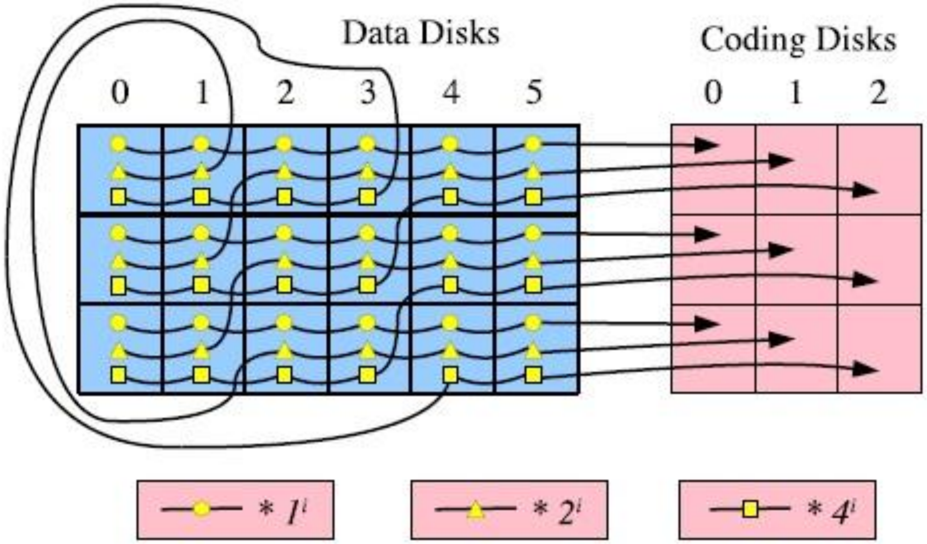
- Coding symbols can be calculated by

$$\text{for } 0 \leq j < 3, \; c_{j,0} = \sum_{i=0}^{k-1} (2^j)^i d_{i,0}$$

USENIX FAST 2012

Rotated Reed-Solomon Codes

k = 6 m = 3 r = 3



- Coding symbols calculated by

$$c_{j,b} = \sum_{i=0}^{\frac{kj}{m}-1} (2^j)^i d_{i,(b+1)\%r} + \sum_{i=\frac{kj}{m}}^{k-1} (2^j)^i d_{i,b}$$

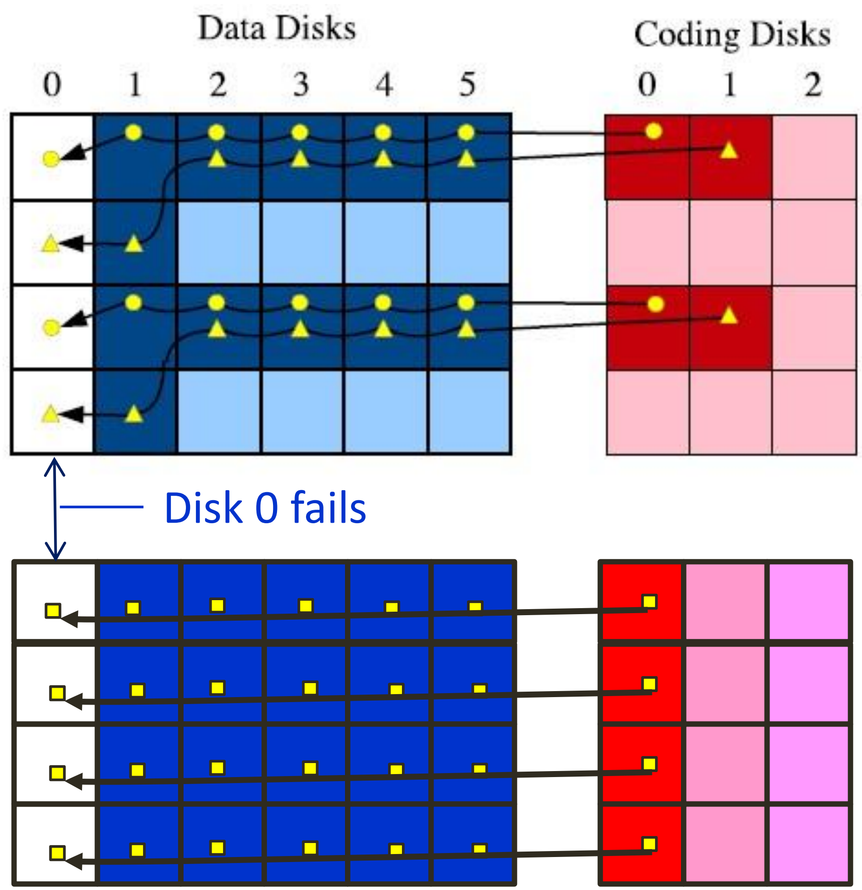
Reconstruction example with Rotated RS Codes

Rotated Reed-Solomon

16 symbols read

P-Drive

24 symbols read



Data symbol read

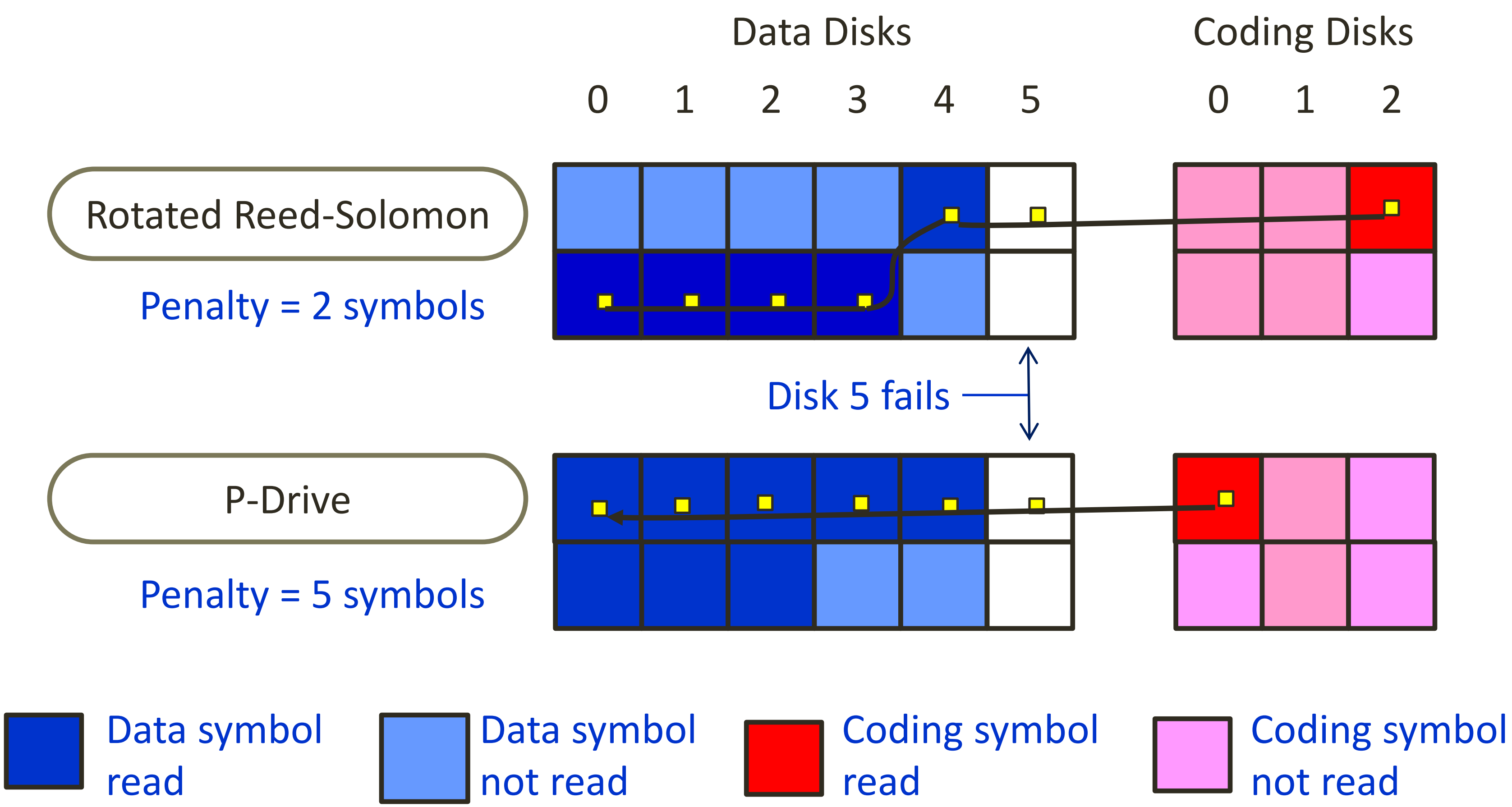
Data symbol not read

Coding symbol read

Coding symbol not read

Degraded Read example with Rotated RS Codes

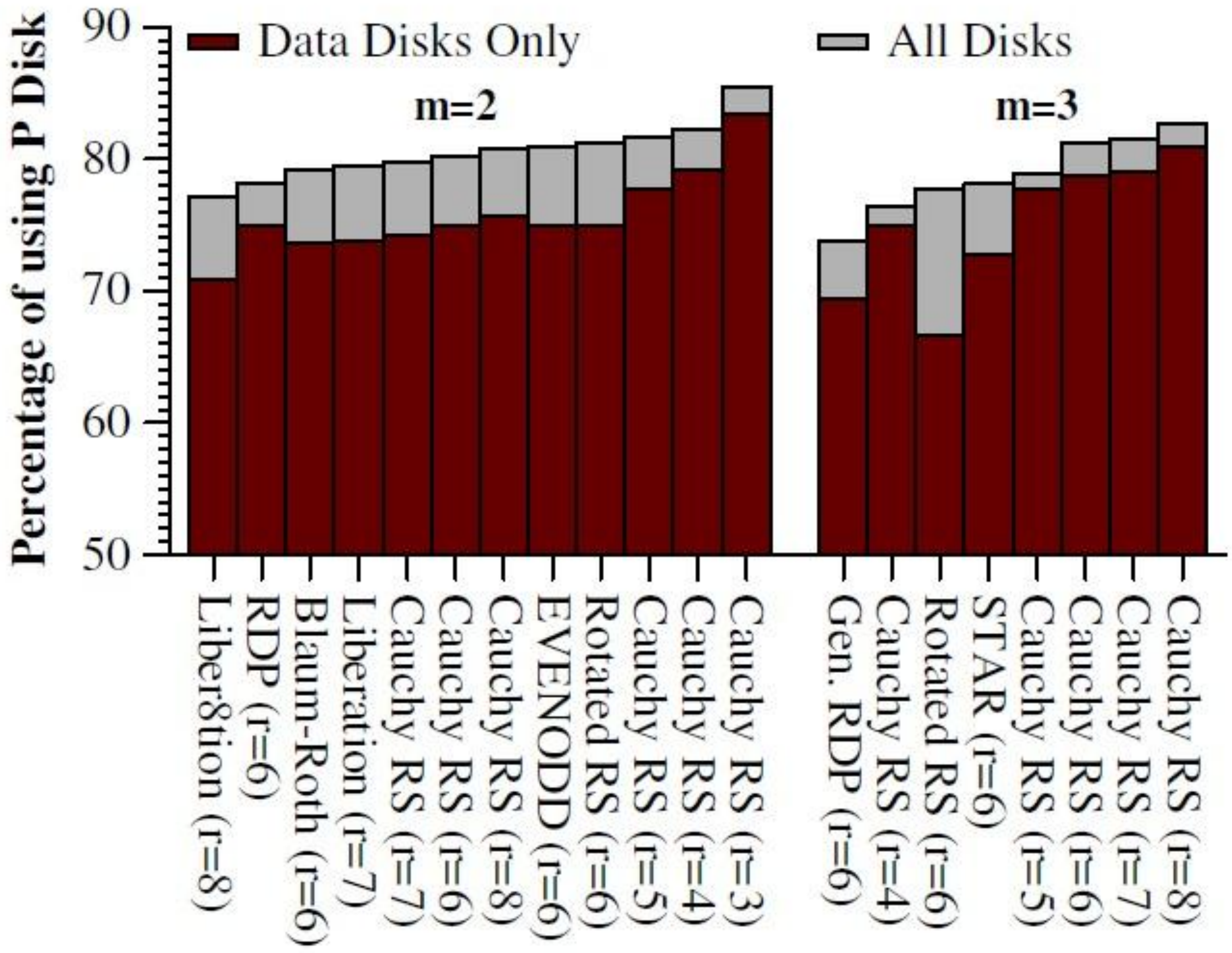
- Read request of 4 symbols starting from $d_{5,0}$
- Penalty = # of symbols read in addition to read request



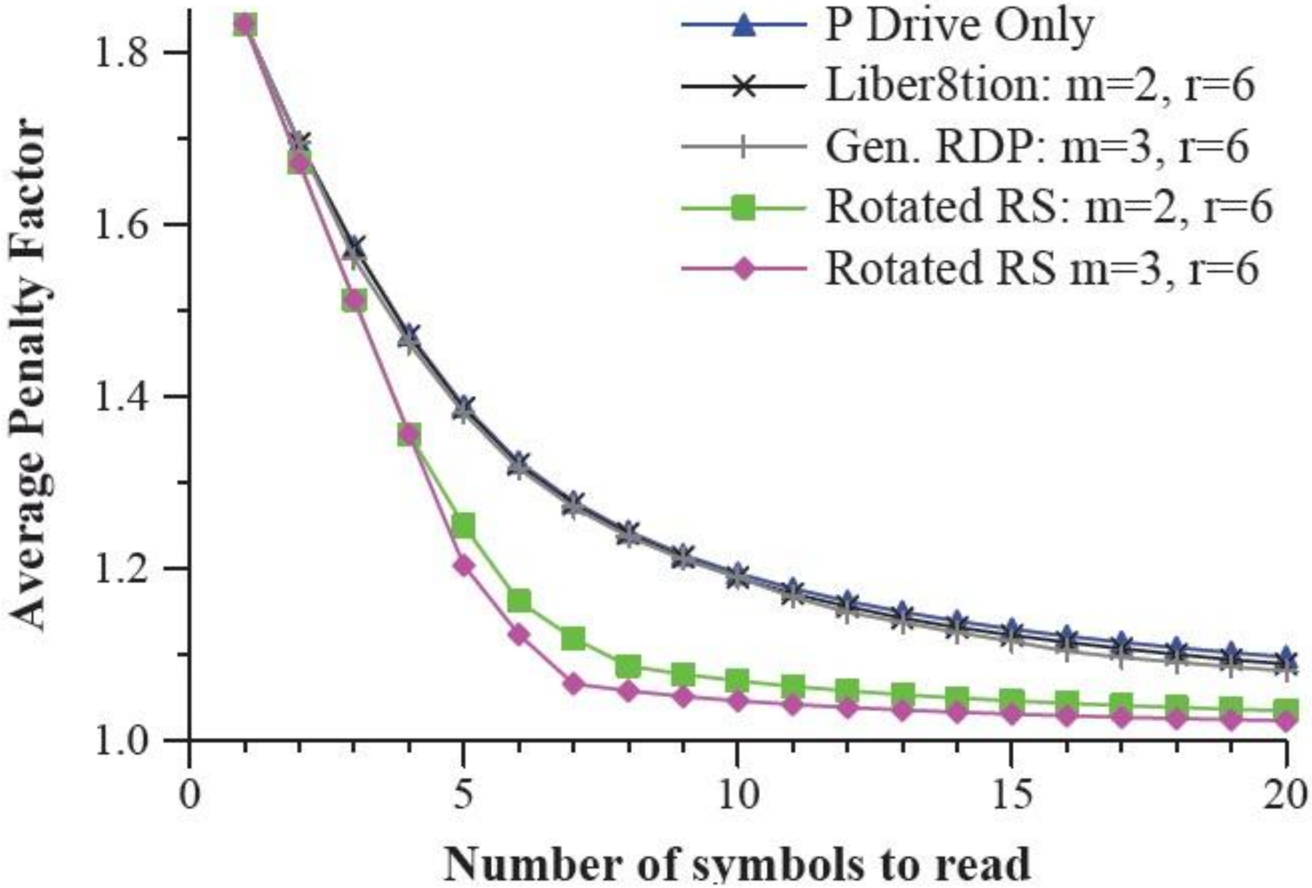
Outline

- Erasure Coded Storage Systems
- Algorithm for minimizing number of symbols
- Rotated Reed-Solomon Codes
- **Analysis & Evaluation**
- Conclusions

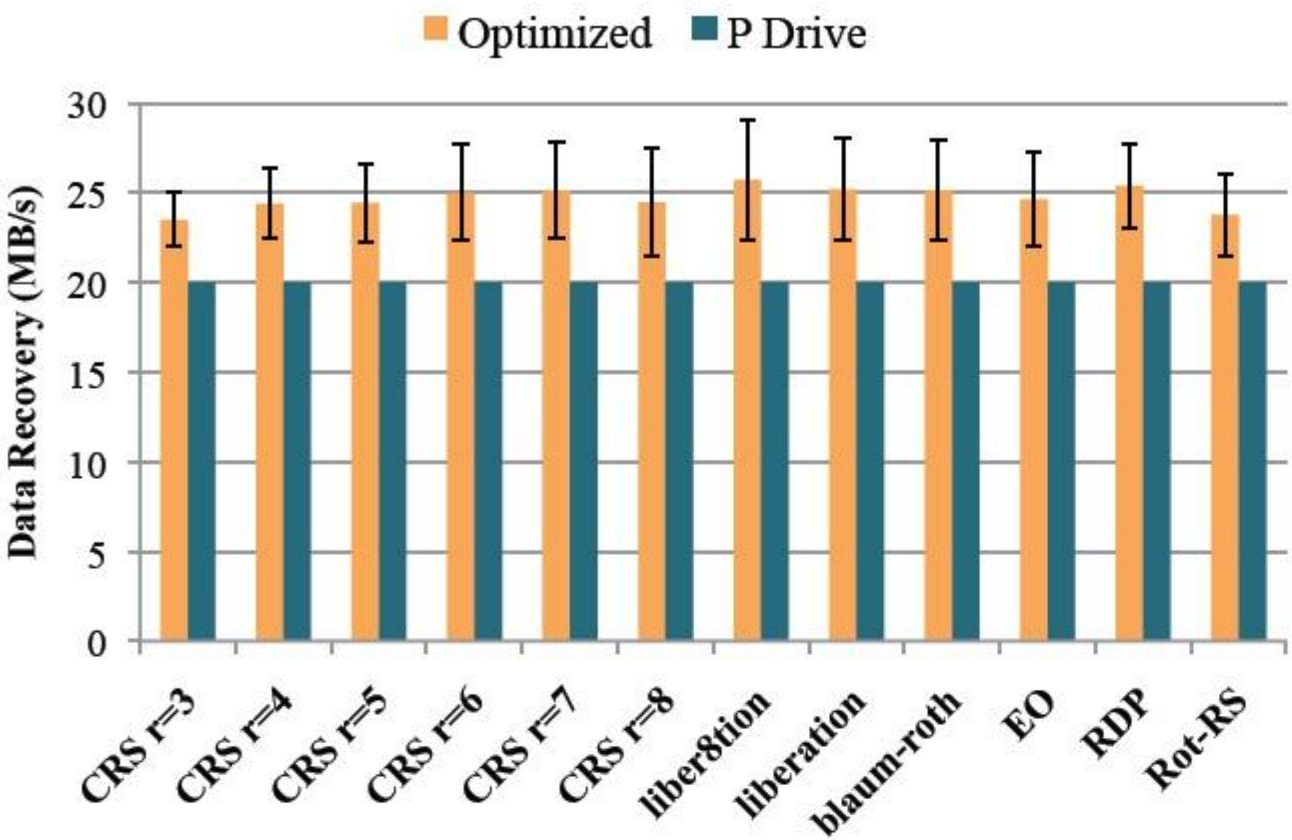
Analysis of Reconstruction



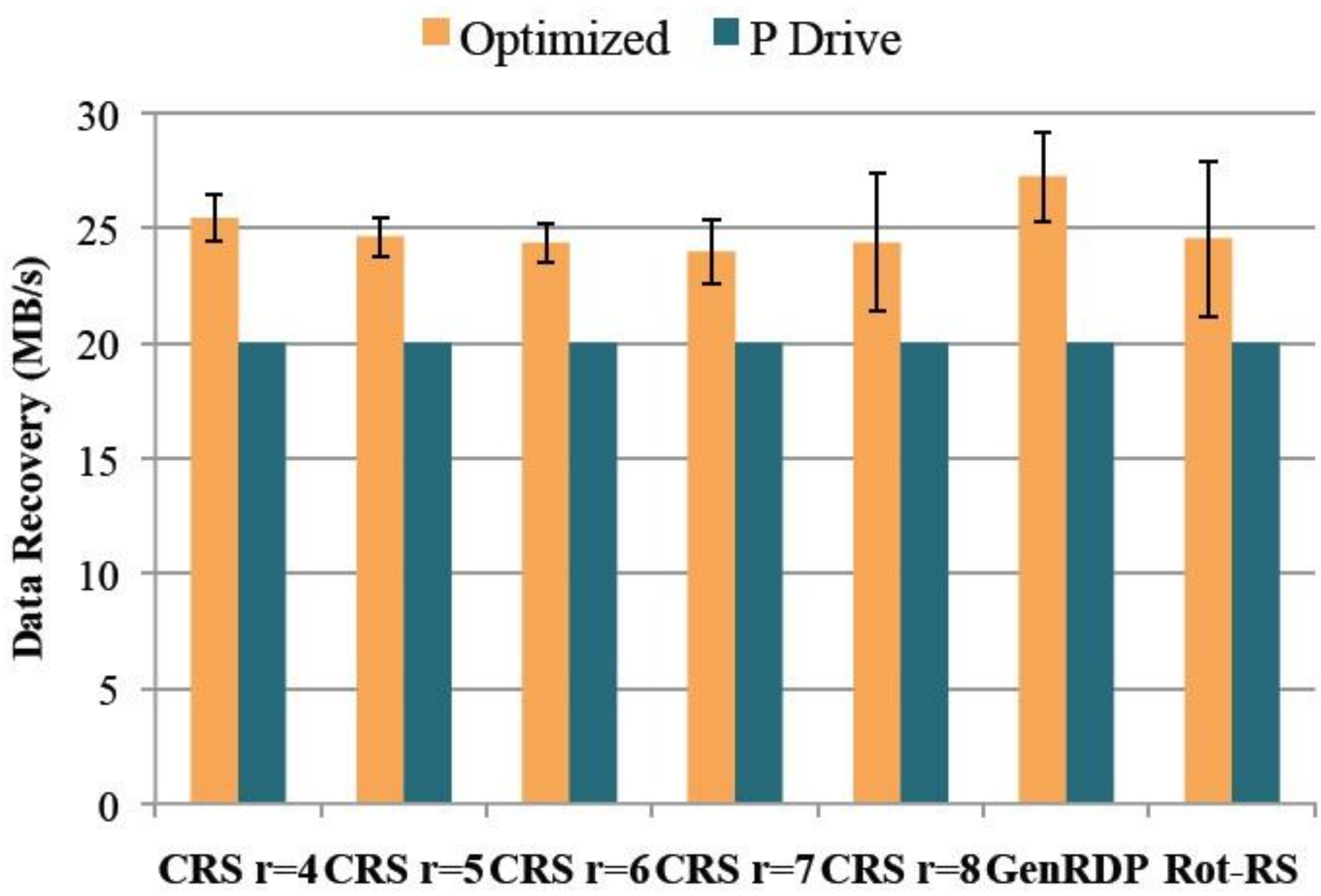
Analysis of Degraded Reads



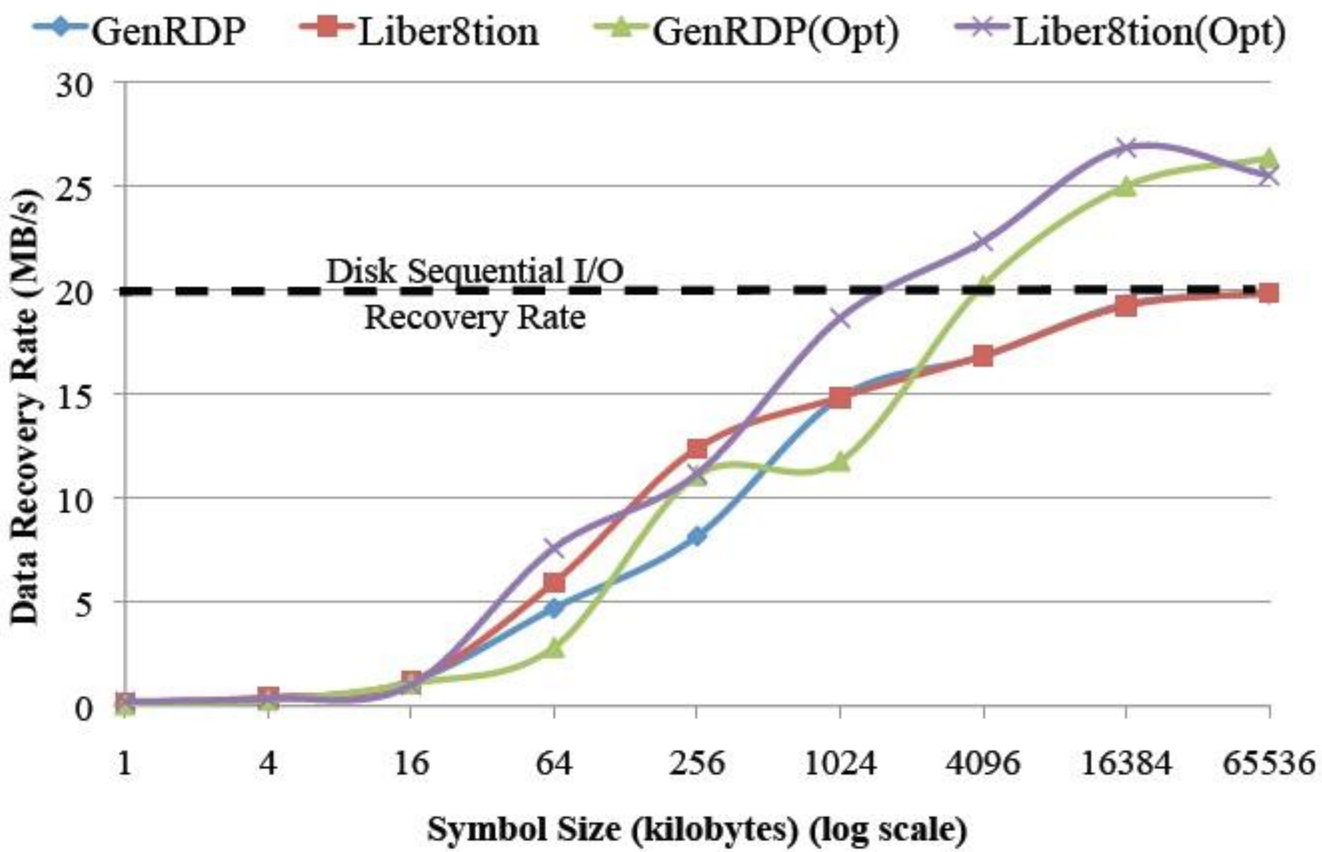
Evaluation of Disk Reconstruction (m = 2)



Evaluation of Disk Reconstruction (m = 3)



The Need for Large Symbols



Outline

- Erasure Coded Storage Systems
- Algorithm for minimizing number of symbols
- Rotated Reed-Solomon Codes
- Analysis & Evaluation
- **Conclusions**

Conclusions

- Traditional RAID based configurations do not give good recovery performance with cloud based erasure coded storage systems
  - Large sealed blocks recommended (at least around 100 MB, preferably > 500 MB)
- Minimizing the number of symbols needed for recovery does result in lower I/O cost
- Generally, optimally-sparse and minimum-density codes perform best for disk reconstruction
- Rotated Reed-Solomon Codes are a better alternative to standard Reed-Solomon for cloud storage

Thank you!