Convertible codes:
Adaptive coding for large-scale data storage

Rashmi Vinayak
Associate Professor
Computer Science Department
Carnegie Mellon University
Cloud storage: Large-scale clusters

- Large scale: Exabytes of data stored on hundreds of thousands to millions of disks
- Failures are common
  - Disk failures measured as annualized failure rates (AFR)
  - AFR => expected % of disk failures in a year
- Erasure codes employed to add redundancy for fault tolerance
Notation and terminology

- **[n, k] code**
  - Encodes k “message” symbols into n “code symbols”
  - “Length” = n and “Dimension” = k
  - Systematic code
  - $r = n - k$ (number of parity symbols)

- Meeting certain decodability requirements
  - Maximum Distance Separable (MDS) = any k out of n sufficient to decode
Erasure coding in distributed storage systems

Erasure coding example: \((n=9, k=6)\) code

\[
\begin{array}{cccccc}
a & b & c & d & e & f \\
\end{array}
\]

\[
\begin{array}{cccccc}
a & b & c & d & e & f & P1 & P2 & P3 \\
\end{array}
\]

data blocks

parity blocks

distributed on disks across servers (across failure domains)
Erasure coding in distributed storage systems

[9, 6] erasure code (6 data, 3 parities)
Redundancy configuration in storage systems

• Amount of redundancy
  - Function of the erasure code parameters, “n” and “k”
  - Example (n=9, k=6): 1.5x redundancy

• Chosen to meet durability, availability, performance requirements
  - Mean time to data loss (MTTDL) target for disk failure rate
  - Reconstruction latency constraints for degraded reads
Redundancy configuration in storage systems

- Code parameters decide: amount of redundancy and fault tolerance
- Chosen to meet durability (e.g. MTTDL) and availability requirements
- Based on (average) failure rate across disk fleet

Today’s redundancy configuration mechanisms are “one-scheme-for-all disks”. However…

[9, 6] erasure code (6 data, 3 parities)
Disk failure rates vary across makes/models

- > 5.3 million HDDs
- > 60 makes/models
- Deployed in production at Google, NetApp, Backblaze

Orders of magnitude variation in failure rate across makes/models

Disk failure rates vary over time with age

Disk hazard (bathtub) curve

<table>
<thead>
<tr>
<th>Age of disk</th>
<th>AFR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infancy</td>
<td>0</td>
</tr>
<tr>
<td>3-5 months</td>
<td>0</td>
</tr>
<tr>
<td>3-4 years</td>
<td>Lower failure rate</td>
</tr>
<tr>
<td>Wearout</td>
<td>14</td>
</tr>
</tbody>
</table>

Reality: different disks fail differently

- Single storage cluster may have multiple makes/models of different ages
Opportunity to reduce storage overhead

Disk-Adaptive Redundancy (DARE)

lower failure rate → lower redundancy → lower storage cost

Savings via adaptive coding

From evaluation on production cluster data at Google and Backblaze

• Potential for 11-16% savings in storage space
  - Translates to 1000s of fewer disks
  - Savings of millions of dollars

• Significant savings due to the scale

Need for Adaptive Coding in Storage Systems

1. Disk failure rates are variable

2. Data temperature varies over time
   - On hot data
     ▪ use low rate, shorter block length, higher redundancy
   - On cold data
     ▪ use higher rate, longer block length, lower redundancy
Collaboration with Google on disk-adaptive coding for real-world storage clusters
Code conversion problem

Convert data encoded under \([n^I, k^I]\) initial code \(C^I\) into data encoded under \([n^F, k^F]\) final code \(C^F\)

Data encoded under \(C^I\) (initial configuration) \(\rightarrow\) Data encoded under \(C^F\) (final configuration)

• Same information stored in initial and final configurations but encoded differently

Code conversion problem

- Default approach:
  **Re-encode the data** on disks undergoing failure rate transition

- Requires reading all the data units and computing new parities

- **High cost of conversion**
  - Typically a large number of code conversions at a time
  - Results in highly varying and large spikes of resource consumption
Challenge: Conversion of coded data

Conversion cost estimate on traces from production clusters at Google
Related work

• Specific cases of code conversion:
  - Rashmi et al 2011, Xia et al. 2015, Mousavi et al. 2018, Wu et al. 2020

• Variants of code conversion:

• Regenerating codes: applicable for conversions with fixed dimension (k) and increasing length (n)
A framework to study code conversion: 
Convertible Codes

• To handle change in dimension from $k^I$ to $k^F$
  - consider $M = \text{lcm}(k^I, k^F)$ message symbols

• Conversion takes multiple codewords in the initial configuration to multiple codewords in the final configuration

Convertible codes framework

\[ [n^I, k^I] \text{ code} \quad \rightarrow \quad [n^F, k^F] \text{ code} \]

**Initial Codewords**

\[ n^I \quad k^I \quad n^I \quad k^I \quad \ldots \quad n^I \quad k^I \]

**Final Codewords**

\[ k^F \quad n^F \quad k^F \quad n^F \quad \ldots \quad k^F \quad n^F \]
Convertible codes framework

• Multiple codewords
  => need to specify how to partition message symbols among codewords

• Initial partition ($\mathcal{P}^I$)
  - map message symbols into initial codewords

• Final partition ($\mathcal{P}^F$)
  - map message symbols into final codewords
Example: Code conversion

- \( C^I : [3,2] \Rightarrow C^F : [5,3] \)
- \( M = \text{lcm}(k^I = 2, k^F = 3) = 6 \)

\[
\begin{array}{ccccccc}
a & b & c & d & e & f & \in \mathbb{F}_q^6
\end{array}
\]
Example: Code conversion

- $C^I : [3,2] \Rightarrow C^F : [5,3]$
Example: Code conversion

- \( C^I : [3,2] \Rightarrow C^F : [5,3] \)

\[
\begin{array}{c}
C^I(a,b) \\
a \quad b
\end{array} \quad \begin{array}{c}
C^I(c,d) \\
c \quad d
\end{array} \quad \begin{array}{c}
C^I(e,f) \\
e \quad f
\end{array} \quad \in \mathbb{F}_q^6
\]

- For systematic codes

\[
\begin{array}{c}
C^I(a,b) : \square \square \square \\
C^I(c,d) : \square \square \square \\
C^I(e,f) : \square \square \square
\end{array} \quad \text{conversion} \quad \begin{array}{c}
\square \square \square \square \square : C^F(a,b,c) \\
\square \square \square \square \square : C^F(d,e,f)
\end{array}
\]
Convertible Codes

Definition $[(n^I, k^I; n^F, k^F)$ Convertible Code$]$

A $(n^I, k^I; n^F, k^F)$ convertible code over $\mathbb{F}_q$ is defined by:

1. a pair of codes $(C^I, C^F)$ over $\mathbb{F}_q$
   - $C^I$ is an $[n^I, k^I]$ code; $C^F$ is an $[n^F, k^F]$ code

2. a pair of partitions $\mathcal{P}^I$, $\mathcal{P}^F$ of $[M = \text{lcm}(k^I, k^F)]$
   - Each subset in $\mathcal{P}^I$ is of size $k^I$ and each subset in $\mathcal{P}^F$ is of size $k^F$

3. a conversion procedure $\mathcal{T}_{C^I \rightarrow C^F}$
Code conversion: Toy example

Initial codeword 1

\[\begin{bmatrix} 5 \\ 3 \end{bmatrix}\]

Initial codeword 2

\[\begin{bmatrix} 8 \\ 6 \end{bmatrix}\]

Final codeword

Data is kept unchanged

Parities can be designed for conversion

\[\begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \\ 1 & 4 \\ 1 & 5 \end{bmatrix}\]
Types of code conversions

General regime

Generalized merge
- Merge

Generalized split
- Split
Code conversion regimes

Merge regime

\[ k^F = \lambda^I k^I \]

Split regime

\[ k^I = \lambda^F k^F \]
Cost of code conversion

1. Access cost
   (# of nodes read or written)

2. Conversion bandwidth
   (total size of data transmitted)
Cost of code conversion for linear MDS codes

Access cost

- Merge regime
- Split regime
- General regime

Conversion bandwidth

- Merge regime
- Split regime

Tight lower bounds + explicit optimal constructions
Theorem

If $r^F \leq r^I$, access cost is $\lambda^I r^F$

If $r^F > r^I$, access cost is $\lambda^I k^I$ (no savings)

How to construct codes here?

$(r = n - k = \text{number of parities for a systematic code})$
Constructing codes

\[
\begin{bmatrix}
A & B & C \\
1 & 1 & 1 \\
\end{bmatrix}
\]

systematic matrix

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 2 & 3 \\
\end{bmatrix}
\]

parity matrix $P$
Properties for efficient conversion

Super-regular + Block-constructible = Optimal MDS convertible code
Systematic + MDS $\iff$ super-regular

Every square submatrix is invertible
Block-constructible

Each block of $P^F$ is spanned by $r^F$ columns of $P^I$
Theorem (informal)
Construction of access-optimal convertible codes for all merge regime parameters for large enough field sizes.
Access cost: general construction

\((k^I = 3, r^I = 3) \rightarrow (k^F = 6, r^F = 3)\)

\[
P^I = \begin{bmatrix}
1 & 1 & 1 \\
1 & \theta & \theta^2 \\
1 & \theta^2 & \theta^4
\end{bmatrix}
\]

\[
P^F = \begin{bmatrix}
1 & 1 & 1 \\
1 & \theta & \theta^2 \\
1 & \theta^2 & \theta^4 \\
1 & \theta^3 & \theta^6 \\
1 & \theta^4 & \theta^8 \\
1 & \theta^5 & \theta^{10}
\end{bmatrix}
\]

\(\theta: \) primitive element

Conversion process

Requires high field size to ensure super-regularity
Theorem (informal)
Low-field size constructions of access-optimal convertible codes for merge regime parameters when $r^F < r^I$. 
Access cost: low field-size construction

- \((k^I = 4, r^I = 3) \rightarrow (k^F = 8, r^F = 2)\)
- Idea: use super-regular Hankel-form array

\(T_{12}:\)

\[
P^I \in \mathbb{F}^{4 \times 3}_{13}
\]

\[
P^F \in \mathbb{F}^{8 \times 2}_{13}
\]
Cost of code conversion for linear MDS codes

Access cost

- Merge regime
  - MR’20
- Split regime
  - MMR’20
- General regime
  - MMR’20

Conversion bandwidth

- Merge regime
  - MR’21
- Split regime
  - MR’22

Tight lower bounds (for merge only) + explicit constructions
• Lower access cost already gives lower conversion bandwidth as well

• Can we achieve further reduction in conversion bandwidth over access-optimal convertible codes?
MERGE REGIME
Bandwidth: lower bound

Variable capacity links
Bandwidth: lower bound

**Theorem**

If \( r^F \leq r^I \),

BW cost is \( \lambda^I r^F \alpha \)

Same as access cost!

If \( r^F > r^I \),

BW cost is

\[
\lambda^I k^I \alpha - \lambda^I r^I \alpha \left( \frac{k^I}{r^F} - 1 \right)
\]

\( \alpha \): vector size

**Relative savings**
Building blocks of code conversion

General regime

Generalized merge

Merge

Generalized split

Split
General conversion

• Via generalized merges and generalized splits
• Example: \([n^I = 6, k^I = 5] \Rightarrow [n^F = 13, k^F = 12]\)
General conversion

- Via generalized merges and generalized splits
- Example: \([n^I = 6, k^I = 5] \Rightarrow [n^F = 13, k^F = 12]\)
• Via generalized merges and generalized splits
• Example: \([n^I = 6, k^I = 5] \Rightarrow [n^F = 13, k^F = 12]\)
Summary

- Code conversion problem
- Convertible codes: A general framework for study of code conversion
- Two metrics of conversion cost: Access and Bandwidth
- Tight lower bounds for certain parameter regimes
- Explicit optimal constructions for certain parameter regimes
- High potential for real-world impact
- BITS Magazine article in the upcoming special issue on storage: "Code Conversions in Storage Systems"
Open problems

- Lower bounds and optimal constructions for general parameter regime

- Bounds on field size and practical (low field size) constructions for all parameters

- Optimizing for conversion simultaneously with other properties
  - Repair (some recent work), update complexity

- Chain conversions and multiple target parameters (some recent work)

Thanks! Questions?