Coding Theory in the time of Cloud Storage

Myna Vajha

Assistant Professor, EE Department

IIT Hyderabad

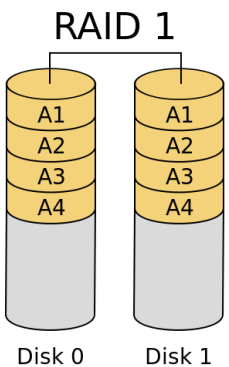
April 18, 2025

Outline

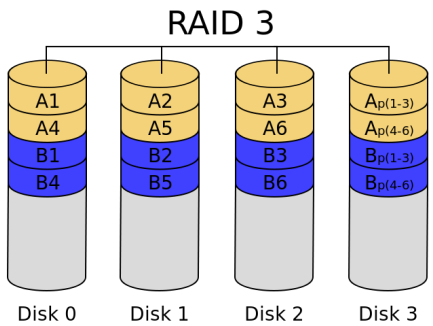
- Erasure Codes and Distributed Storage.
- Node repair problem
  1. Regenerating Codes
     - Coupled Layer(Clay) code
  2. Other Approaches
- Code conversion problem
- Overview of applications of coding theory

How to save yourself from Data Loss ?

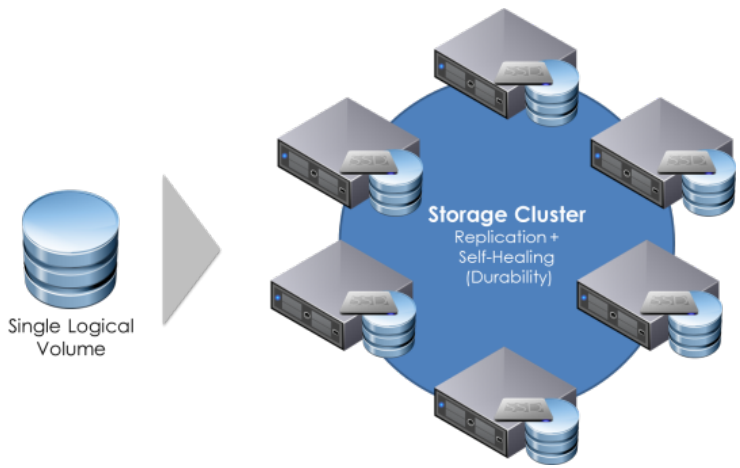
1. Redundant Array of Independant Disks (RAID)
   - RAID 1: uses replication
   - RAID 3: adds a parity
2. Distributed Storage (Replication or Erasure Codes)



2-Replication



$$A_{p(1-3)} = A_1 \oplus A_2 \oplus A_3$$



DSS

Image Souces: Wikipedia, vifx.co.nz, techmissile.com

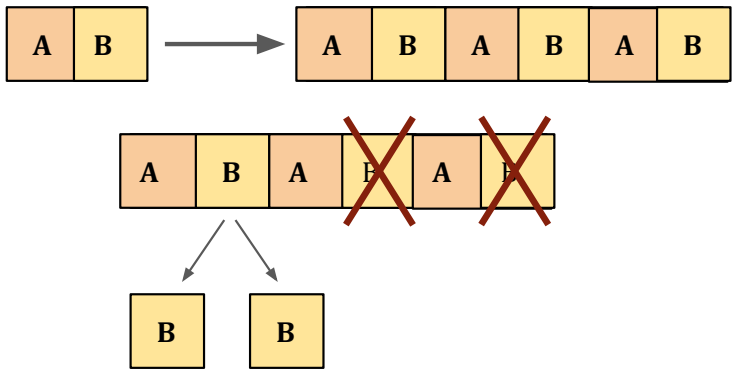
Erasure Code

- Used to recover from data loss.
- In a (n, k) erasure code, k units of data are encoded to get n units of data.
- A Maximum Distance Seperable (MDS) Code provides reliability against erasure of any $r = n - k$ units.
  - *Examples*: RS(Reed Solomon), RAID6.
- Storage Overhead of such a code $\nu = \frac{n}{k} = 1 + \frac{r}{k}$

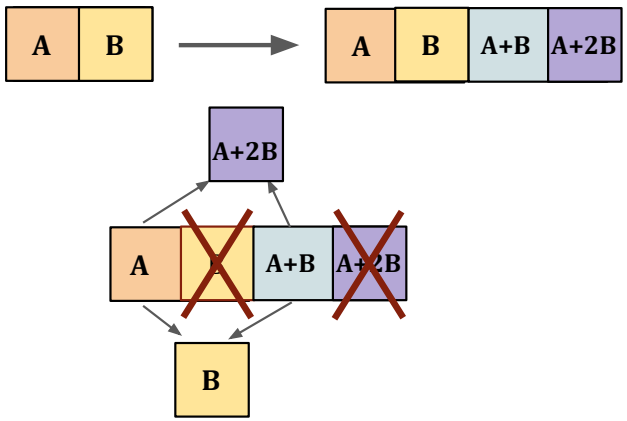
- Above $(6, 4)$ MDS code has storage overhead 1.5 and it can recover from 2 erasures.

Replication Code vs Erasure Code

- Google's GFS uses 3-replication code.
- Facebook's Hadoop uses [10,4] Reed Solomon(RS) Code.



(6,2) 3-Replication

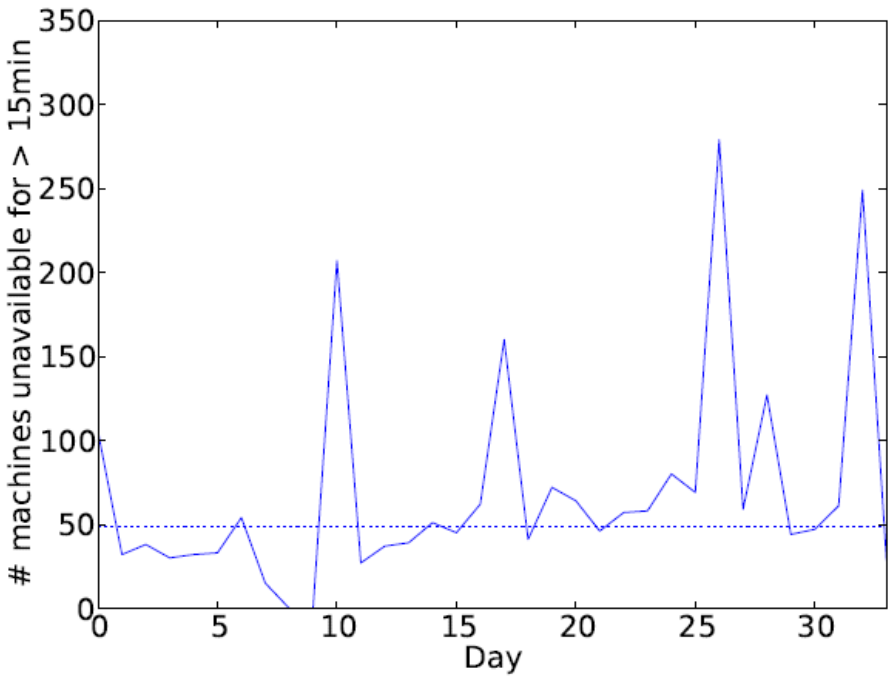


[4,2] Reed Solomon

Code	Storage O/h	Bandwidth [1]	Reliability
[6,2] 3-rep	3x	0.5	2
[4,2] RS	2x	1.0	2

[1] As a fraction of k units

Erasure Codes and Node Failures



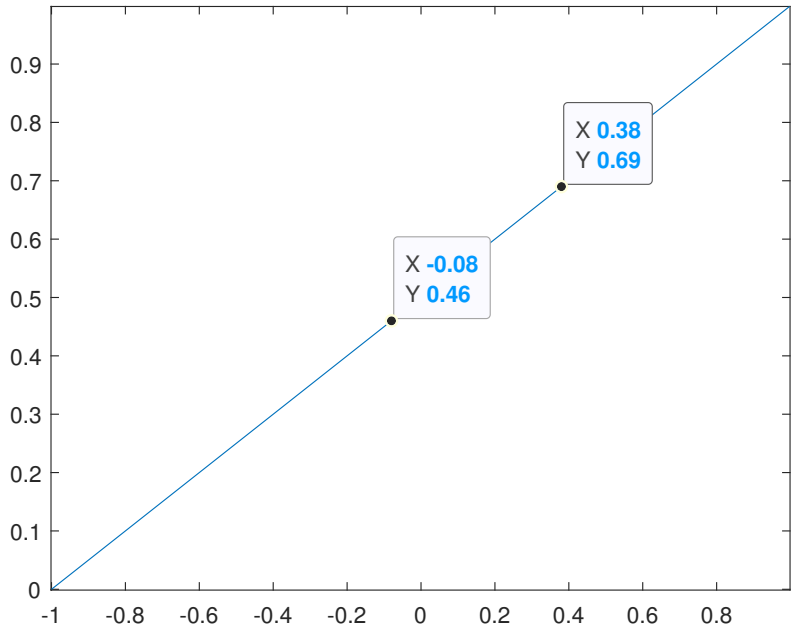
- A median of 50 nodes are unavailable per day.
- 98% of the failures are single node failures.
- A median of 180TB of network traffic per day is generated in order to reconstruct the RS coded data corresponding to unavailable machines.
- **Thus there is a strong need for erasure codes that can efficiently recover from single-node failures.**

Image courtesy: Rashmi et al.: "A Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster," USENIX Hotstorage, 2013.

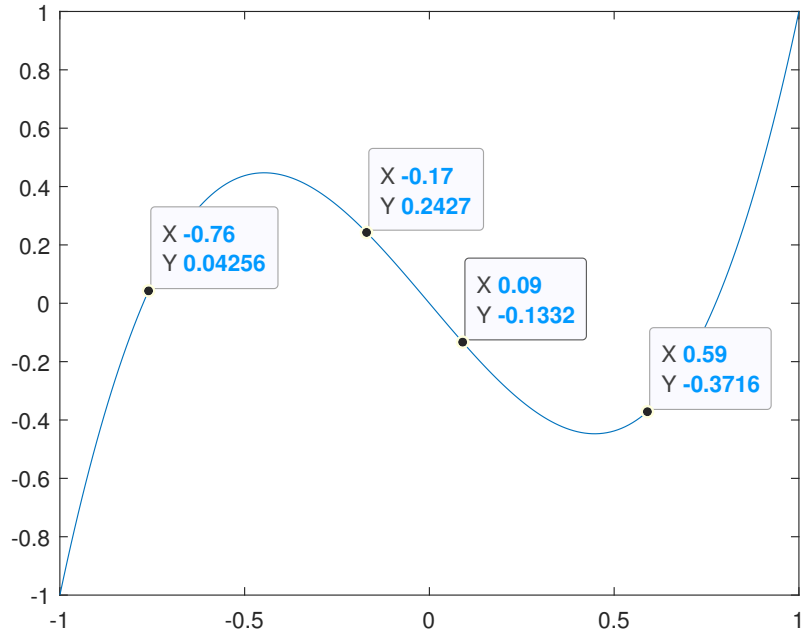
Node Repair Problem

Conventional Node Repair of an RS Code

- For an $[n, k]$ RS code. A message of k symbols can be thought of as coefficients of a polynomial with degree $< k$.



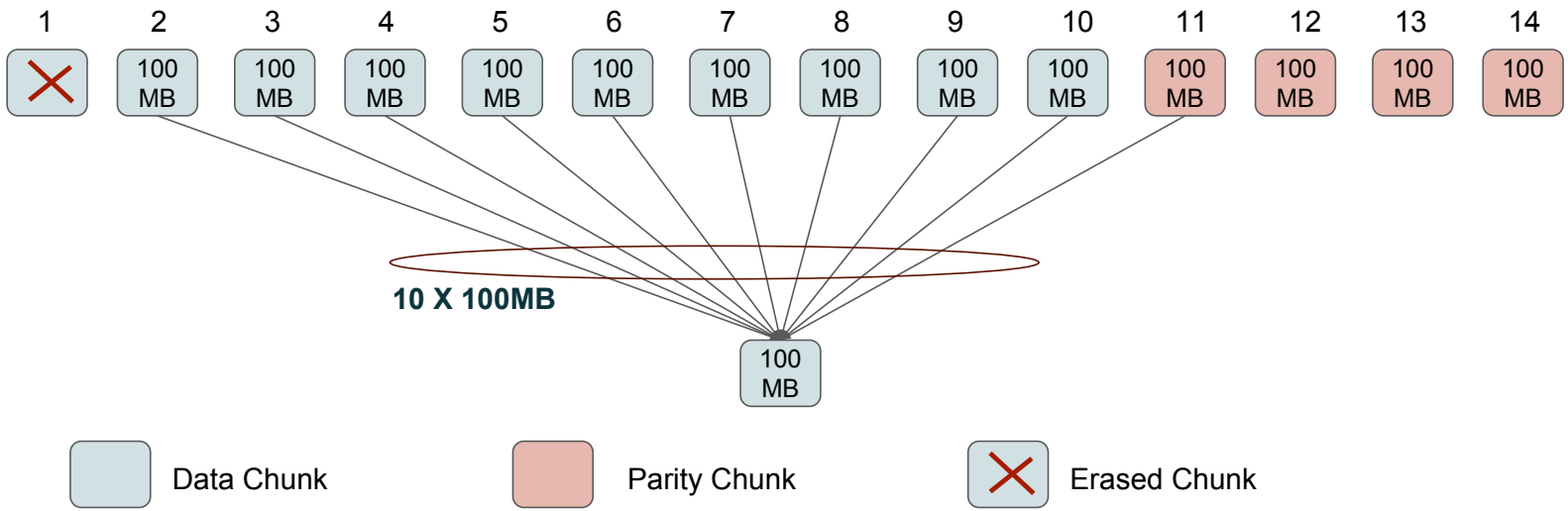
$k = 2$, the polynomial is a line. Two evaluations enough to recover a line



$k = 4$, the polynomial is of degree 3. Four evaluations enough to recover it

- If you have one evaluation of a line, then are infinitely many options for the lines that pass through the point.
- Need k evaluations to recover any polynomial of degree $< k$.

Conventional Node Repair of an RS Code



In the example $(14, 10)$ RS code,

1. the amount of data downloaded to repair 100MB of data equals 1GB.

clearly, there is room for improvement...

Things that matter.

- Low Storage Overhead. ✓
- High Tolerance for erasures. ✓
- Fast Repair (Single Node Failure). ??
  - Computation efficiency.
  - Speed in procuring repair data.
    - low repair traffic.
    - smaller disk read latencies.

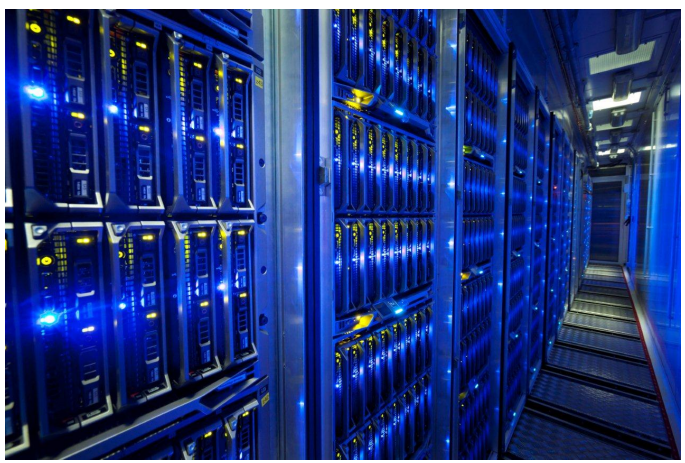
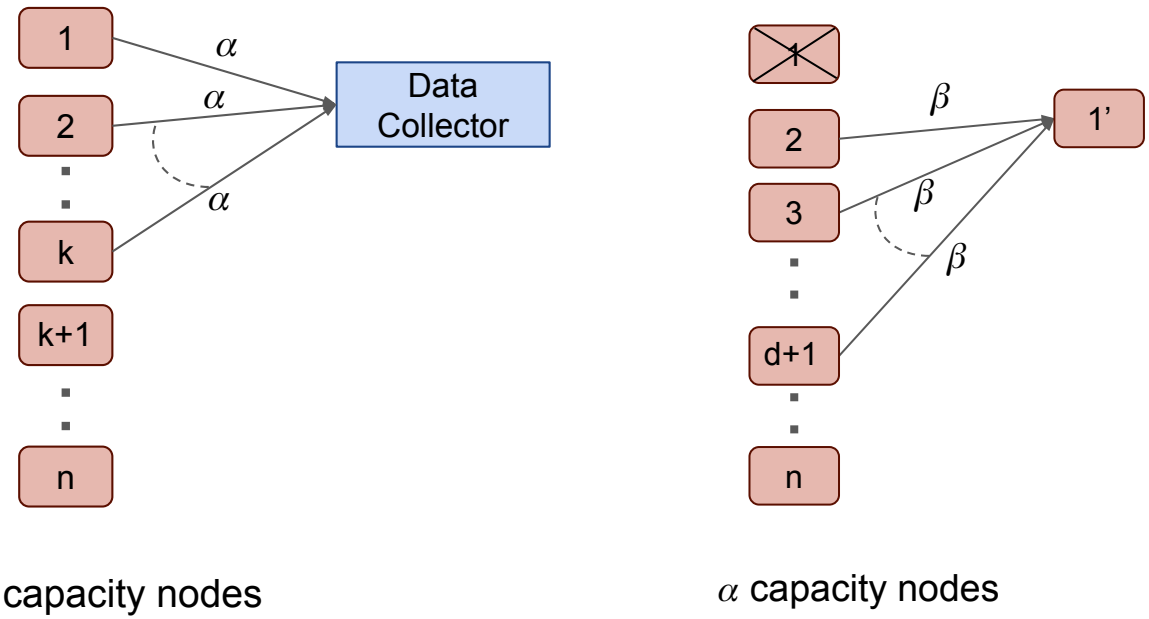


Image source: sine.co, cartoonstock.com

Regenerating Codes

Parameters: $(\ (n,k,d),\ (\alpha,\beta),\ B,\ \mathbb{F}_q\ )$

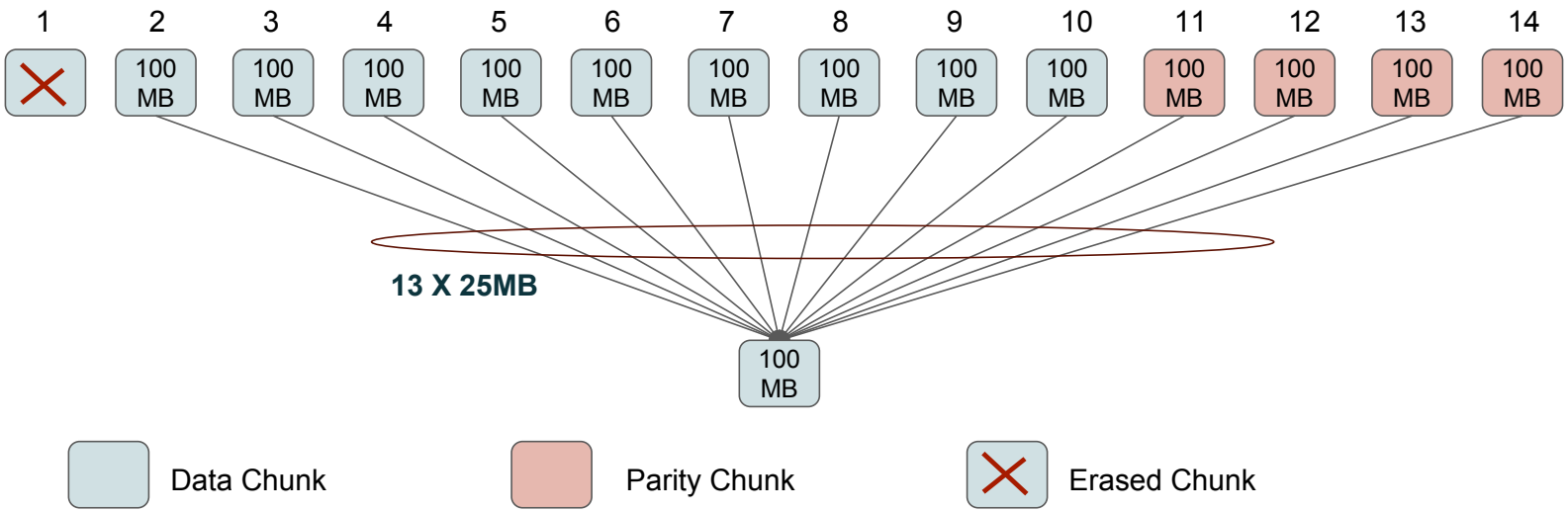


- Data (of size B) can be recovered by connecting to any k of n nodes
- A failed node can be repaired by connecting to any d nodes, downloading β symbols from each node; ($d\beta << $ file size B)

A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems", IEEE Transactions on Information Theory, 2010

Minimum Storage Regenerating Code

- MSR codes are MDS that have least possible repair bandwidth



1. Size of failed node's contents: 100MB
2. RS repair BW: 1 GB
3. MSR Repair BW: 325 MB

4-way Optimality of Clay code

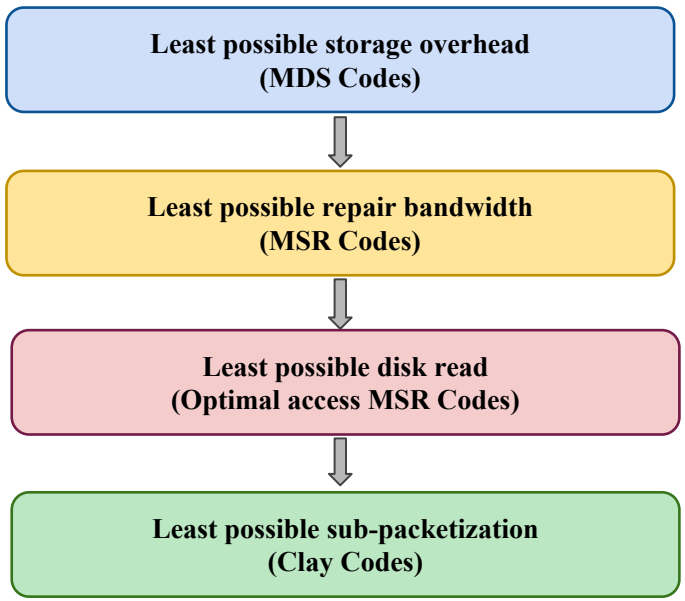
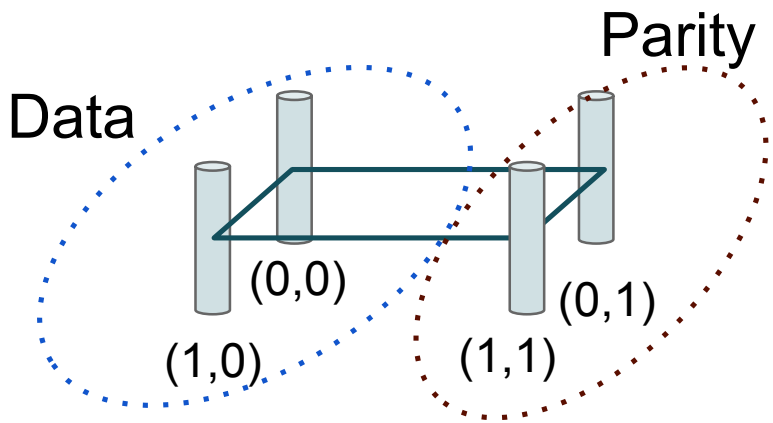


Image courtesy: denverpost.com

Moulding an MDS Code to Yield the Clay Code

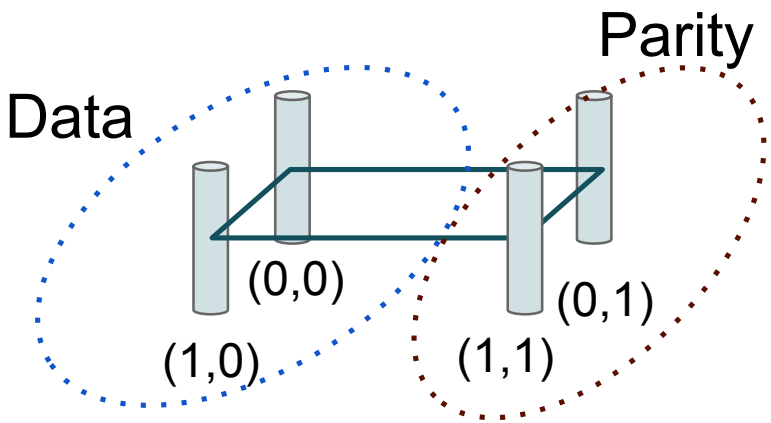
$(n = 4, k = 2)$ MDS code with optimal repair of systematic nodes, $\alpha = 2$



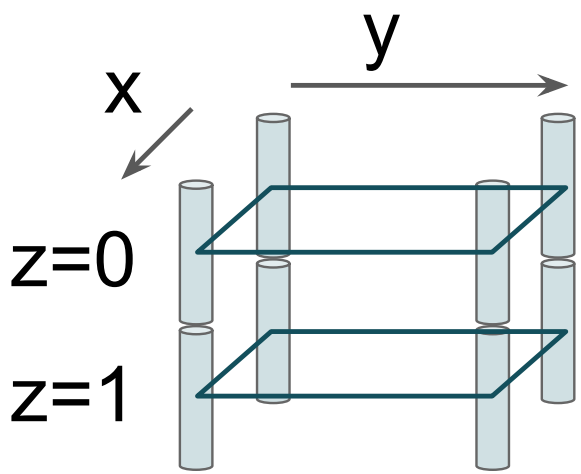
Code symbols of $[4, 2]$ MDS code.

Moulding an MDS Code to Yield the Clay Code

$(n = 4, k = 2)$ MDS code with optimal repair of systematic nodes, $\alpha = 2$



Code symbols of $[4, 2]$ MDS code.

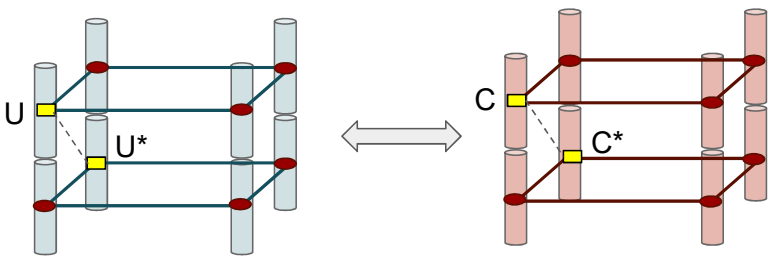


Layer two such units

Uncoupled code still needs 4 symbols during recovery of single node (containing 2 symbols).

Moulding an MDS Code to Yield the Clay Code

$(n = 4, k = 2)$ MDS code with optimal repair of systematic nodes, $\alpha = 2$



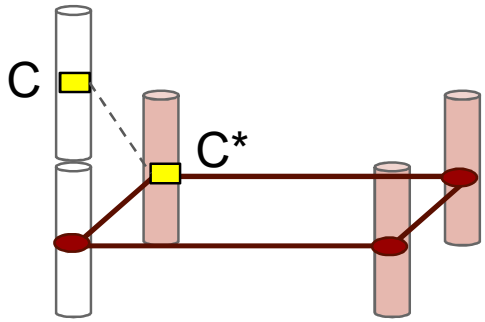
UnCoupled Code Coupled Code

- Uncoupled code has 2 planes, where each plane corresponds to an $[4, 2]$ MDS code
- Coupled code symbols are obtained by:
  - Copying symbols with red dots
  - Pair of yellow symbols $\{C, C^*\}$ are obtained by transformation

$$\begin{bmatrix} C \\ C^* \end{bmatrix} = \begin{bmatrix} 1 & \gamma \\ \gamma & 1 \end{bmatrix} \begin{bmatrix} U \\ U^* \end{bmatrix}$$

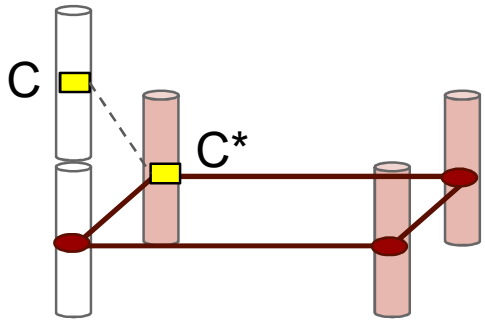
- Note that recovery of any failed node in Uncoupled code requires 4 symbols

Repair from single node loss

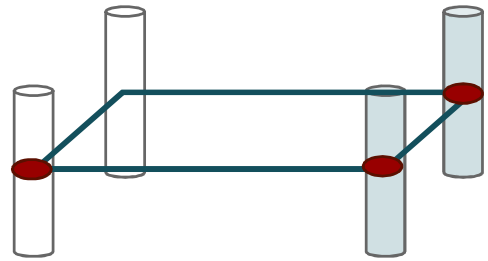


symbols that are available as part of helper information

Repair from single node loss

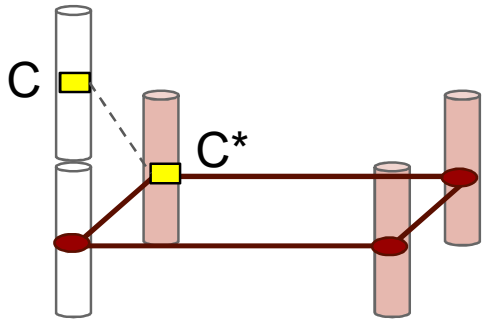


symbols that are available as part of helper information

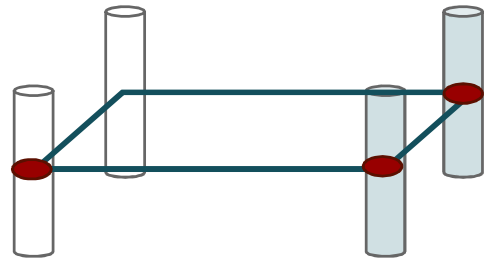


symbols that are computable in uncoupled cube

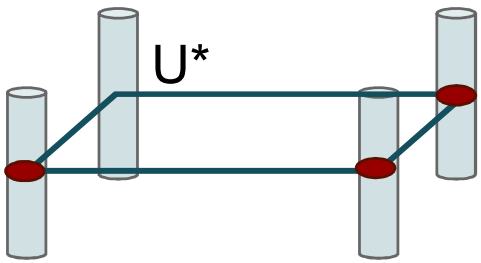
Repair from single node loss



symbols that are available as part of helper information

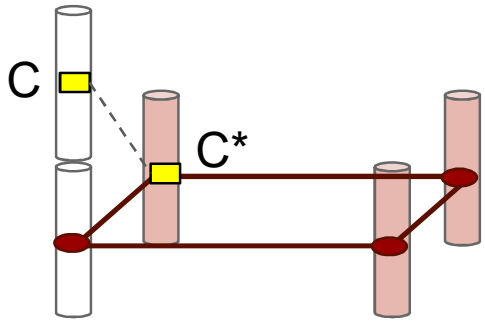


symbols that are computable in uncoupled cube

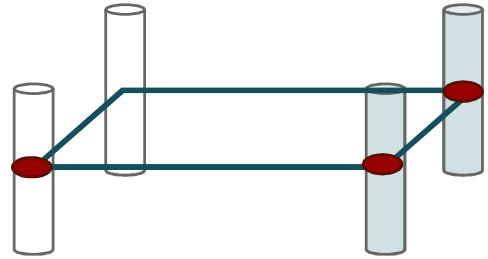


symbols recovered after using the $[4, 2]$ MDS code

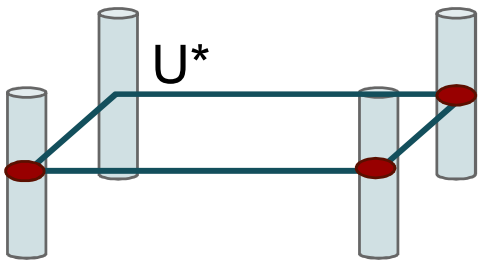
Repair from single node loss



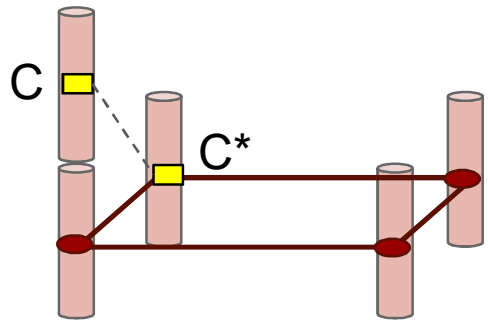
symbols that are available as part of helper information



symbols that are computable in uncoupled cube

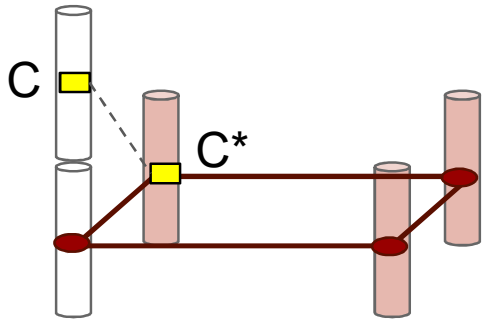


symbols recovered after using the $[4, 2]$ MDS code

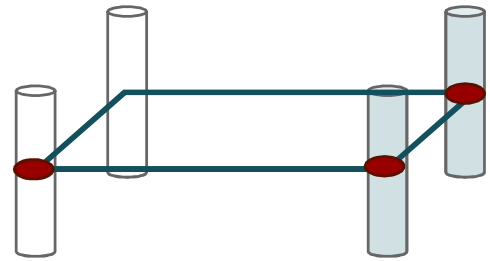


C recovered from C^*, U^*

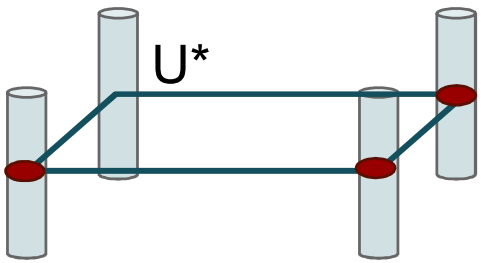
Repair from single node loss



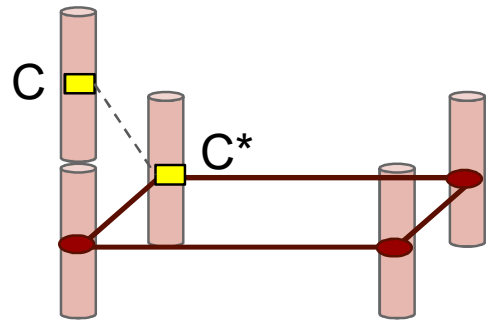
symbols that are available as part of helper information



symbols that are computable in uncoupled cube



symbols recovered after using the $[4, 2]$ MDS code

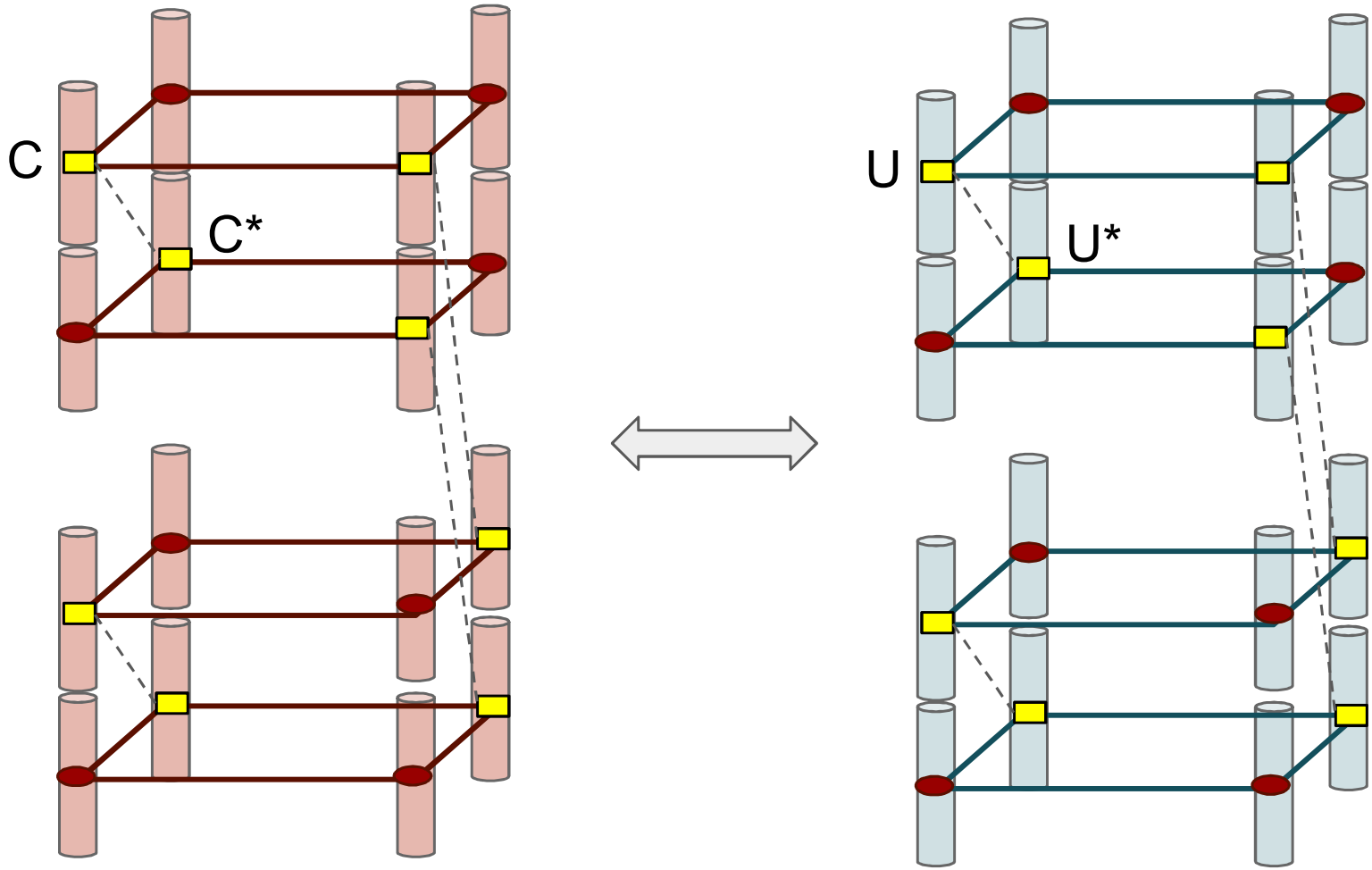


C recovered from C^*, U^*

- Used 3 symbols to recover the 2 symbols. For a file of size 1GB distributed across 2 500MB nodes. Need 750MB to recover the 500MB node instead of the whole 1GB.

Clay Code

$(n = 4, k = 2, d = 3)$ MSR code with all node optimal repair



Coupled Code

Uncoupled Code

- The same construction extends to any (n, k, d)

Ceph: Contributions

A popular opensource distributed storage system used by CERN, Flipkart, Cisco etc

Ceph: Contributions

A popular opensource distributed storage system used by CERN, Flipkart, Cisco etc

"Us (+Vinayak) pitching Clay codes to Ceph in April 2017"

We have introduced Clay code as erasure code plugin. It is part of Ceph's Nautilus release (March 2019). As part of this we also introduced support for vector codes in Ceph.

Vajha et. al., Clay Codes: Moulding MDS Codes to Yield an MSR Code, USENIX FAST 2018.

Clay Code Summary

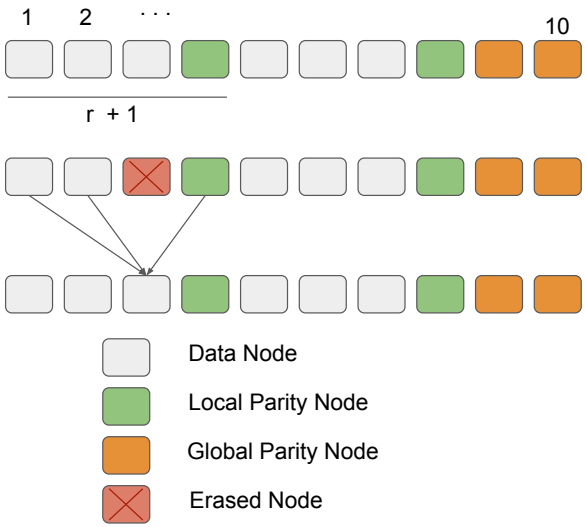
- The open-source implementation of Clay code in Ceph is for any (n, k, d) parameters.

- In comparison to $(20, 16)$ RS code, for Workloads with large sized objects (64MB), the Clay code $(20, 16, 19)$:
  - resulted in repair time reduction by $3X$.
  - Improved degraded read and write performance by 27.17% and 106.68% respectively.

Vajha et. al., Clay Codes: Moulding MDS Codes to Yield an MSR Code, USENIX FAST 2018.

Alternate Approaches for Efficient Node Repair and Availability

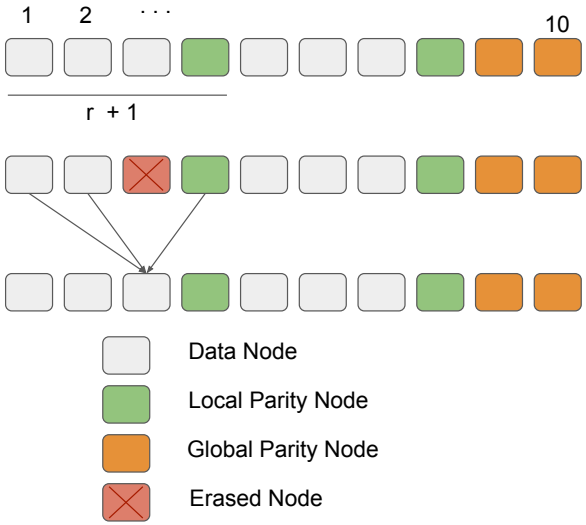
Locally Recoverable Codes (LRC) & Availability



$$\begin{array}{|c|c|c|c|c|c|c|}
\hline
x_{1,1} & x_{1,2} & \cdots & x_{1,j} & \cdots & x_{1,\sigma} & p_1^{(1)} \\
\hline
x_{2,1} & x_{2,2} & \cdots & x_{2,j} & \cdots & x_{2,\sigma} & p_2^{(1)} \\
\hline
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
\hline
x_{i,1} & x_{i,2} & \cdots & x_{i,j} & \cdots & x_{i,\sigma} & p_i^{(1)} \\
\hline
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
\hline
x_{\sigma,1} & x_{\sigma,2} & \cdots & x_{\sigma,j} & \cdots & x_{\sigma,\sigma} & p_\sigma^{(1)} \\
\hline
p_1^{(2)} & p_2^{(2)} & \cdots & p_j^{(2)} & \cdots & p_\sigma^{(2)} & \\
\hline
\end{array}$$

- In a (n, k, r) LRC code, repair of a data node can be done by receiving helper data from r nodes instead of k nodes.

Locally Recoverable Codes (LRC) & Availability



$$\begin{array}{|c|c|c|c|c|c|c|}
\hline
x_{1,1} & x_{1,2} & \cdots & x_{1,j} & \cdots & x_{1,\sigma} & p_1^{(1)} \\
\hline
x_{2,1} & x_{2,2} & \cdots & x_{2,j} & \cdots & x_{2,\sigma} & p_2^{(1)} \\
\hline
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
\hline
x_{i,1} & x_{i,2} & \cdots & x_{i,j} & \cdots & x_{i,\sigma} & p_i^{(1)} \\
\hline
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
\hline
x_{\sigma,1} & x_{\sigma,2} & \cdots & x_{\sigma,j} & \cdots & x_{\sigma,\sigma} & p_\sigma^{(1)} \\
\hline
p_1^{(2)} & p_2^{(2)} & \cdots & p_j^{(2)} & \cdots & p_\sigma^{(2)} & \\
\hline
\end{array}$$

- In a (n, k, r) LRC code, repair of a data node can be done by receiving helper data from r nodes instead of k nodes.
- Shown above is $(10, 6, 3)$ LRC code.
- In the above code each symbol can be recovered using 2-disjoint local parity checks, each involving σ symbols.
- This code has availability 2 and locality σ.

P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the Locality of Codeword Symbols," IEEE Trans. Inf. Theory, vol. 58, no. 11, pp. 6925–6934, Nov. 2012.

Image source: A. Fazeli, A. Vardy, and E. Yaakobi, "PIR with low storage overhead: Coding instead of replication," 2015.

Efficient Repair Of RS codes

1. Classic repair uses k evaluations to recover the polynomial and then recovered the erased symbol
2. Efficient repair of RS
   - consider each evaluation value defined in finite field say, F_2^m as a vector of m symbols.
   - Allows for downloading $< m$ symbols corresponding to each evaluation value
   - can reduce repair bandwidth

- Guruswami and Wootters: Repairing Reed Solomon Codes, ACM, ToC 2019
- Vinayak et.al Codes for distributed storage, Foundations and Trends® in Communications and Information Theory

Code Conversion Problem

MDS Convertible Codes

- Want to change parameters of the code as the data becomes hot/warm/cold to be resource efficient.
- Want initial and final codes to be MDS with params:

$$[n^I, k^I] \rightarrow [n^F, k^F]$$

- Trivial way would be to recover all the message symbols and do a re-encoding of them to new parameters.
- Can one do better ?

MDS Convertible Codes

- Want to change parameters of the code as the data becomes hot/warm/cold to be resource efficient.
- Want initial and final codes to be MDS with params:

$$[n^I, k^I] \rightarrow [n^F, k^F]$$

- Trivial way would be to recover all the message symbols and do a re-encoding of them to new parameters.
- Can one do better ?
- Yes. Convertible codes optimize the access and bandwidth cost
- We recently have some linear field size MDS convertible code constructions based on additive and multiplicative sub-groups of a finite field.

- Maturana, V. S. C. Mukka, K. V. Rashmi, Access-optimal Linear MDS Convertible Codes for All Parameters, ISIT, 2020
- Krishnan, Vajha et. al.: On Linear Field Size Access-Optimal MDS Convertible Codes, (to appear in ISIT 2025)

Other Applications of Coding Theory

- Coding for communication systems
  - Polar codes, LDPC codes, Reed Muller codes
  - Are they capacity achieving, what is the finite block-length performance ?
- Codes for packet level erasure correction
  - Streaming codes with latency guarantees, Fountain, Raptor codes
  - How do they help with real-time applications like gaming, video/audio calls
- Codes for distributed compute
  - matrix-vector multiplication, matrix-matrix multiplication, polynomial function computation, non-linear function computation, distributed ML model training
- Availability codes for privacy, consistency in databases

Thank You!!!

Questions ?