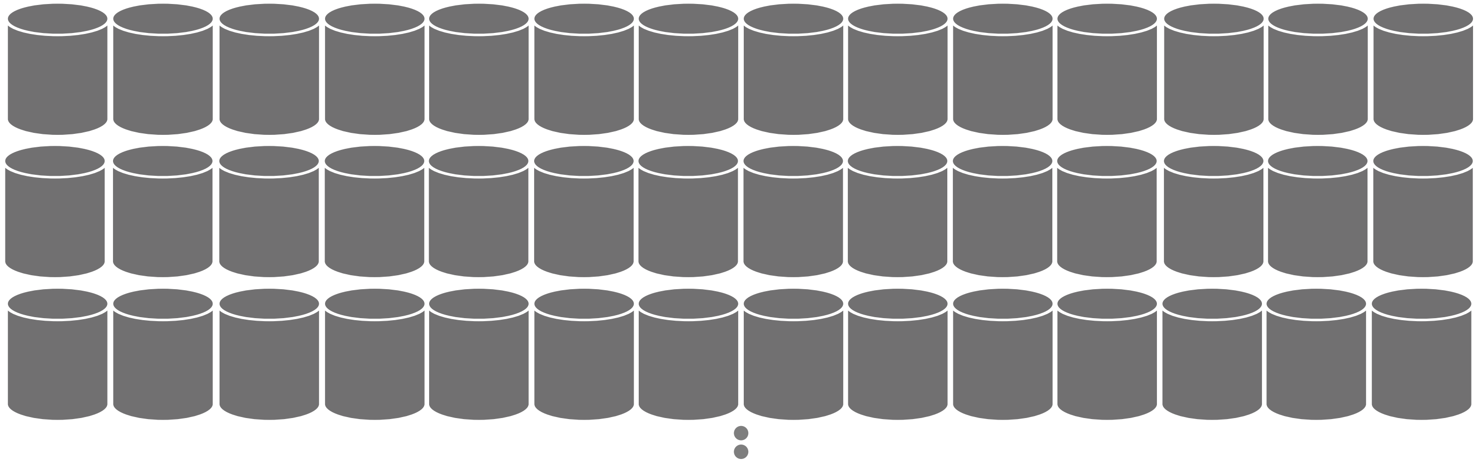
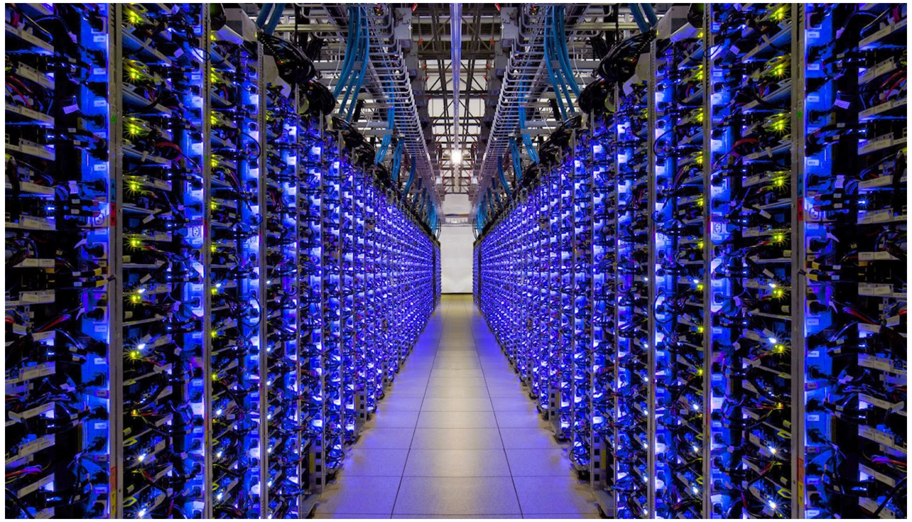
Convertible codes:
Adaptive coding for large-scale data storage

Rashmi Vinayak
Associate Professor
Computer Science Department
Carnegie Mellon University

Cloud storage: Large-scale clusters

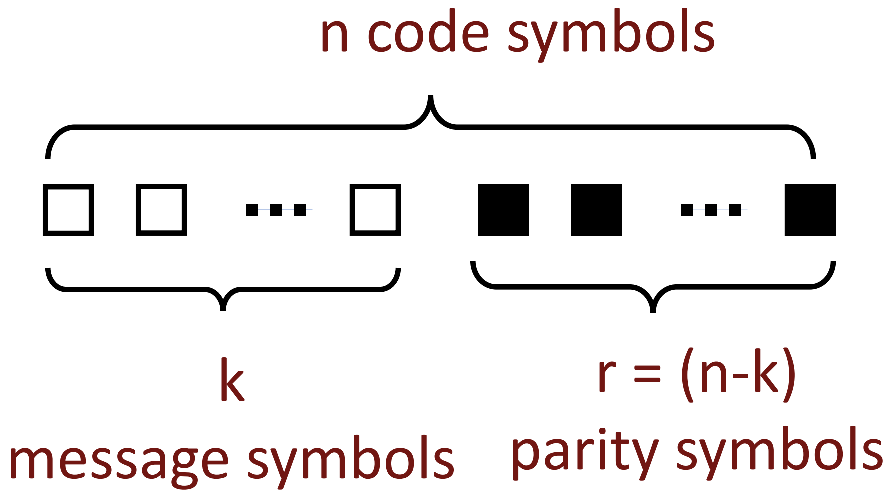




- Large scale: Exabytes of data stored on hundreds of thousands to millions of disks
- Failures are common
  - Disk failures measured as annualized failure rates (AFR)
  - AFR => expected % of disk failures in a year
- Erasure codes employed to add redundancy for fault tolerance

Notation and terminology

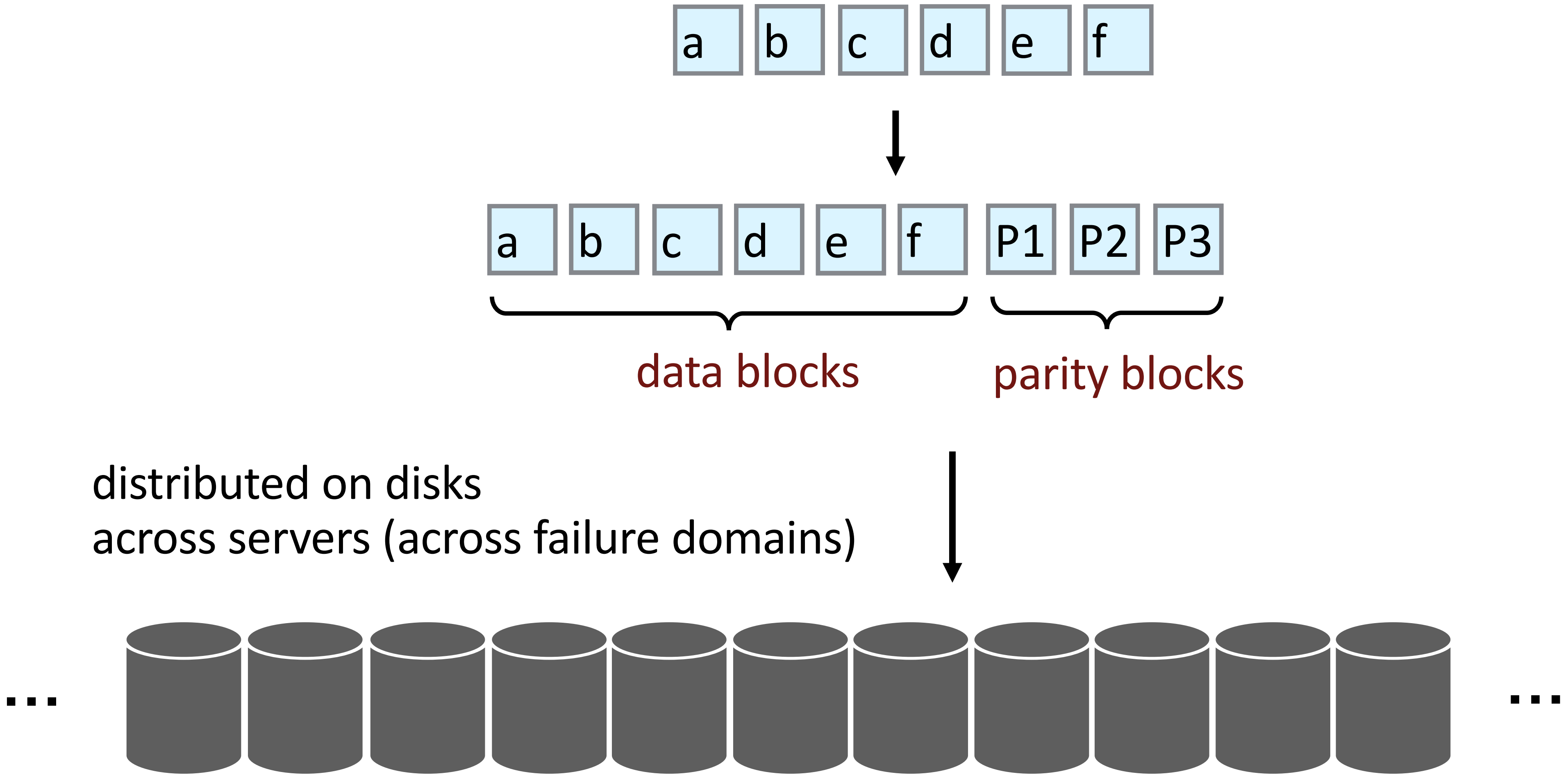
- [n, k] code
  - Encodes k “message” symbols into n “code symbols”
  - “Length” = n and “Dimension” = k
  - Systematic code
  - r = n-k (number of parity symbols)



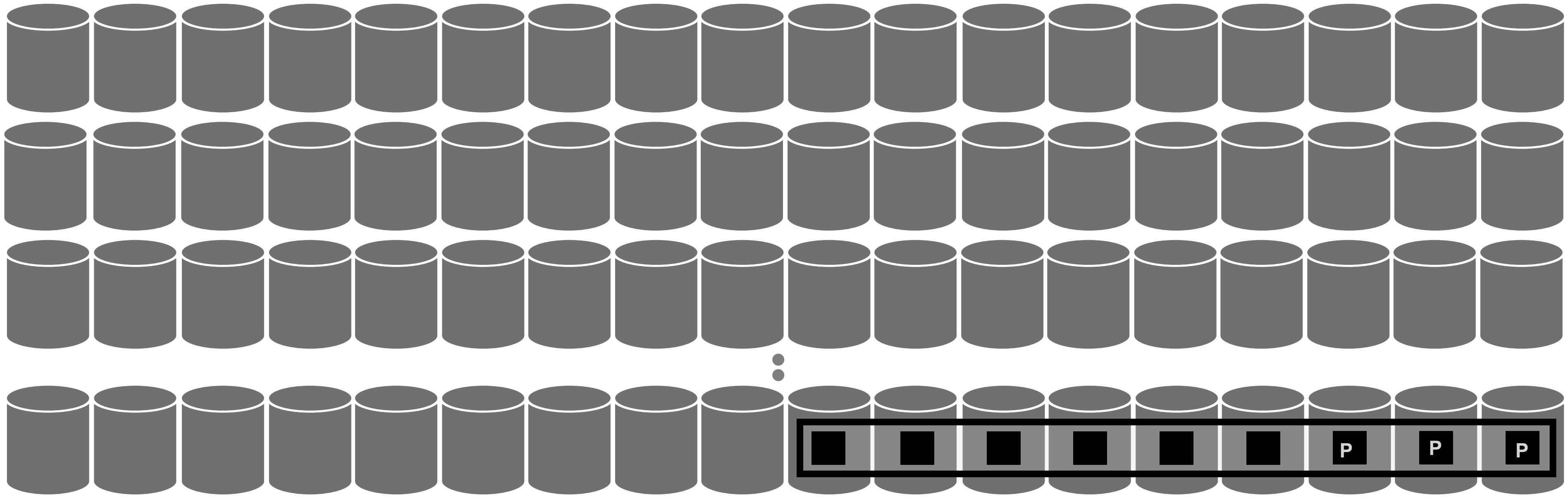
- Meeting certain decodability requirements
  - Maximum Distance Separable (MDS) = any k out of n sufficient to decode

Erasure coding in distributed storage systems

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



Erasure coding in distributed storage systems



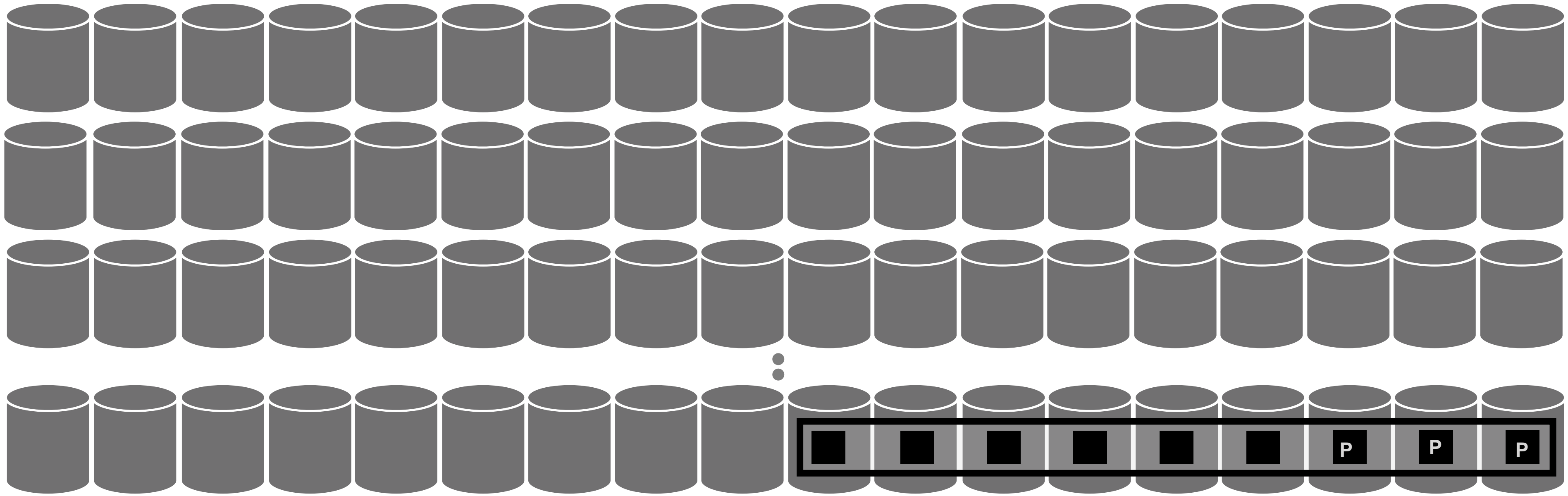
[9, 6] erasure code (6 data, 3 parities)

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

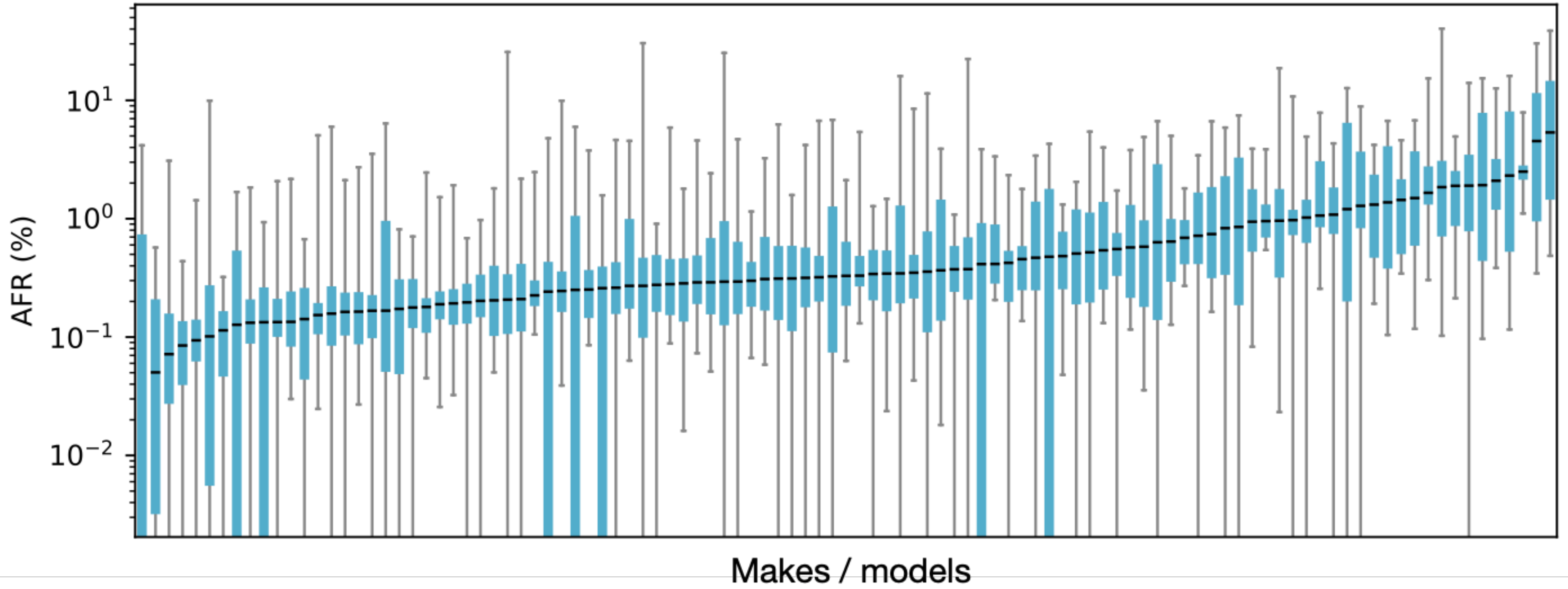


[9, 6] erasure code (6 data, 3 parities)

Today's redundancy configuration mechanisms are "one-scheme-for-all disks".
However…

Disk failure rates vary across makes/models

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

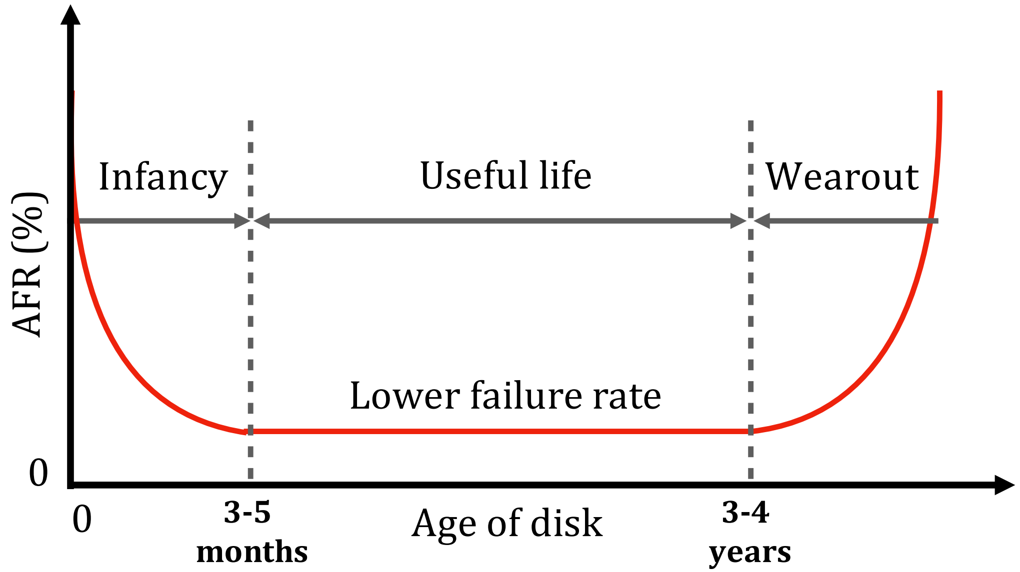


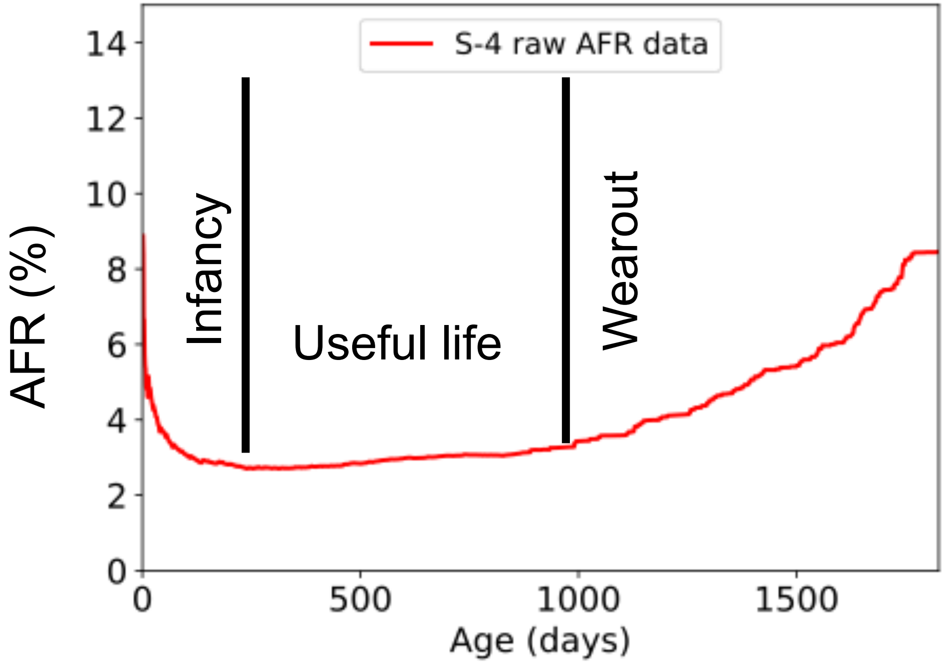
Orders of magnitude variation in failure rate across makes/models

S. Kadekodi, F. Maturana, S. Subramanya, J. Yang, K.V. Rashmi, G. Ganger, "Pacemaker: avoiding HeART attacks in storage clusters with disk-adaptive redundancy", USENIX OSDI, 2020.

Disk failure rates vary over time with age

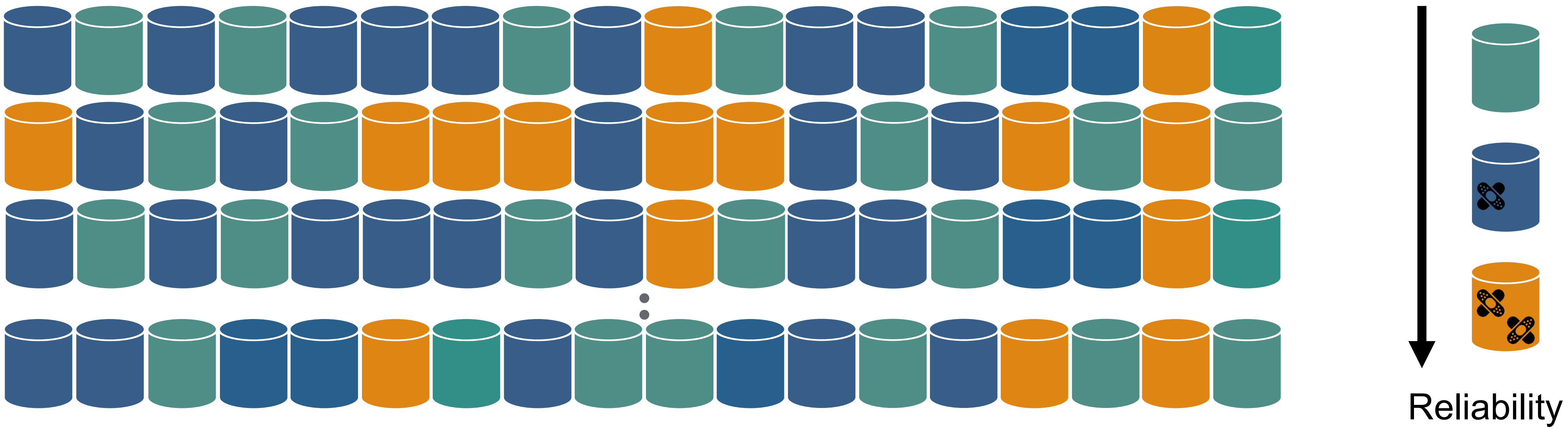
Disk hazard (bathtub) curve





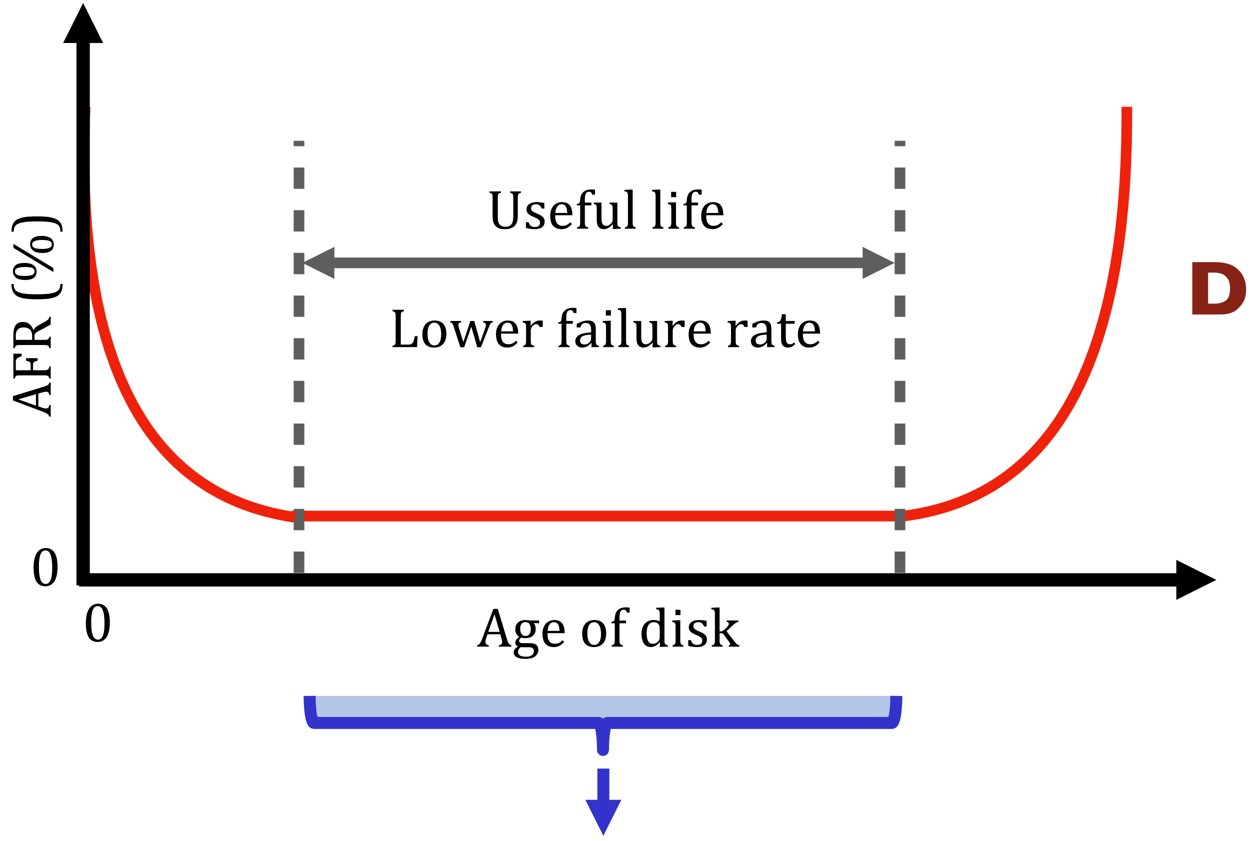
S. Kadekodi, K. V. Rashmi, and G. Ganger, "Cluster storage systems gotta have HeART: improving storage efficiency by exploiting disk-reliability heterogeneity", USENIX FAST 2019.

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

S. Kadekodi, K. V. Rashmi, and G. Ganger, “Cluster storage systems gotta have HeART: improving storage efficiency by exploiting disk-reliability heterogeneity”, USENIX FAST 2019.

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

1. S. Kadekodi, K. V. Rashmi, and G. Ganger, “Cluster storage systems gotta have HeART: improving storage efficiency by exploiting disk-reliability heterogeneity”, USENIX FAST 2019.
2. S. Kadekodi, F. Maturana, S. Subramanya, J. Yang, K.V. Rashmi, G. Ganger, "Pacemaker: avoiding HeART attacks in storage clusters with disk-adaptive redundancy”, USENIX OSDI, 2020.

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

F. Maturana and K.V. Rashmi, "Convertible Codes: Enabling Efficient Conversion of Coded Data in Distributed Storage", ITCS 2020 and IEEE Transactions on Information Theory 2022.

Code conversion problem

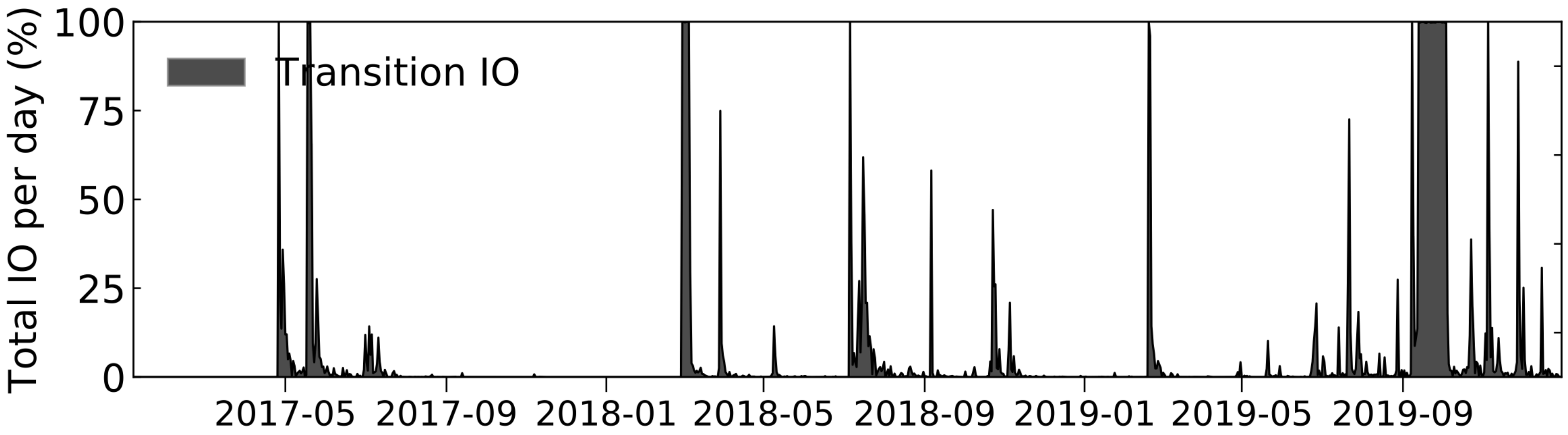
- Default approach:
  Re-encode the data on disks undergoing failure rate transition

- 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**



High cost of conversion

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:
  - Huang et al. 2015, Rai et al 2015, Sonowal & Rai 2017, Hu et al. 2018, Su et al. 2020
- Regenerating codes: applicable for conversions with fixed dimension (k) and increasing length (n)
  - Dimakis et al 2010, El Rouayheb & Ramchandran 2010, Rashmi et al 2011, Shah et al 2011, Suh & Ramchandran 2011, Cadambe et al 2011, Shah et al 2012, Tamo et al 2013, Papailiopoulos et al 2013, Sasidharan et al 2015, Guruswami & Wooters 2016, Ye & Barg 2017, Dau & Milenkovic 2017, Rashmi et al 2017, Chowdhury & Vardy 2018, Hou et al. 2019, Mital et al 2019., Chen & Barg 2019, Mahdaviani et al 2019, Alrabiah & Guruswami 2019, Chen et al. 2020, …

A framework to study code conversion: Convertible Codes

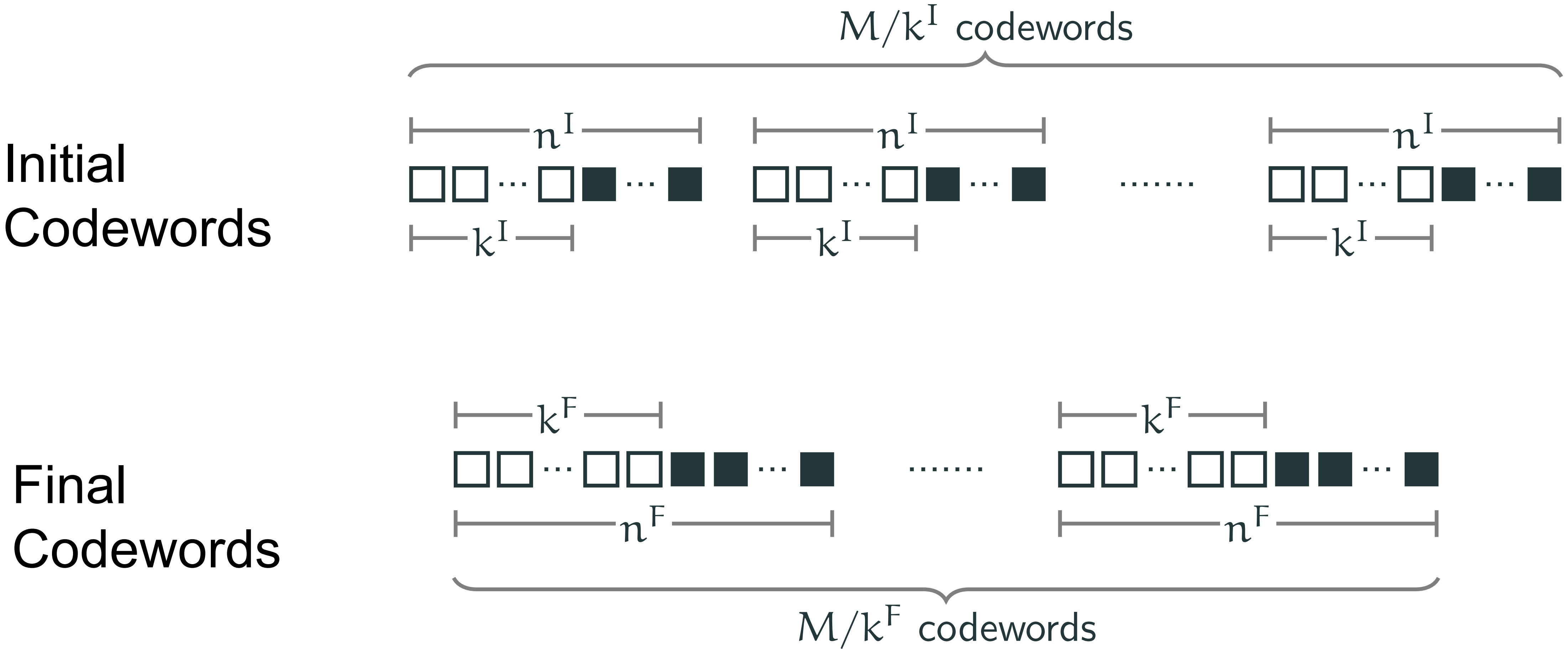
- To handle change in dimension from k^I to k^F
  - consider $M = \mathrm{lcm}(k^I, k^F)$ message symbols

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

F. Maturana and K.V. Rashmi, "Convertible Codes: Enabling Efficient Conversion of Coded Data in Distributed Storage", ITCS 2020 and IEEE Transactions on Information Theory, 2022.

Convertible codes framework

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



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

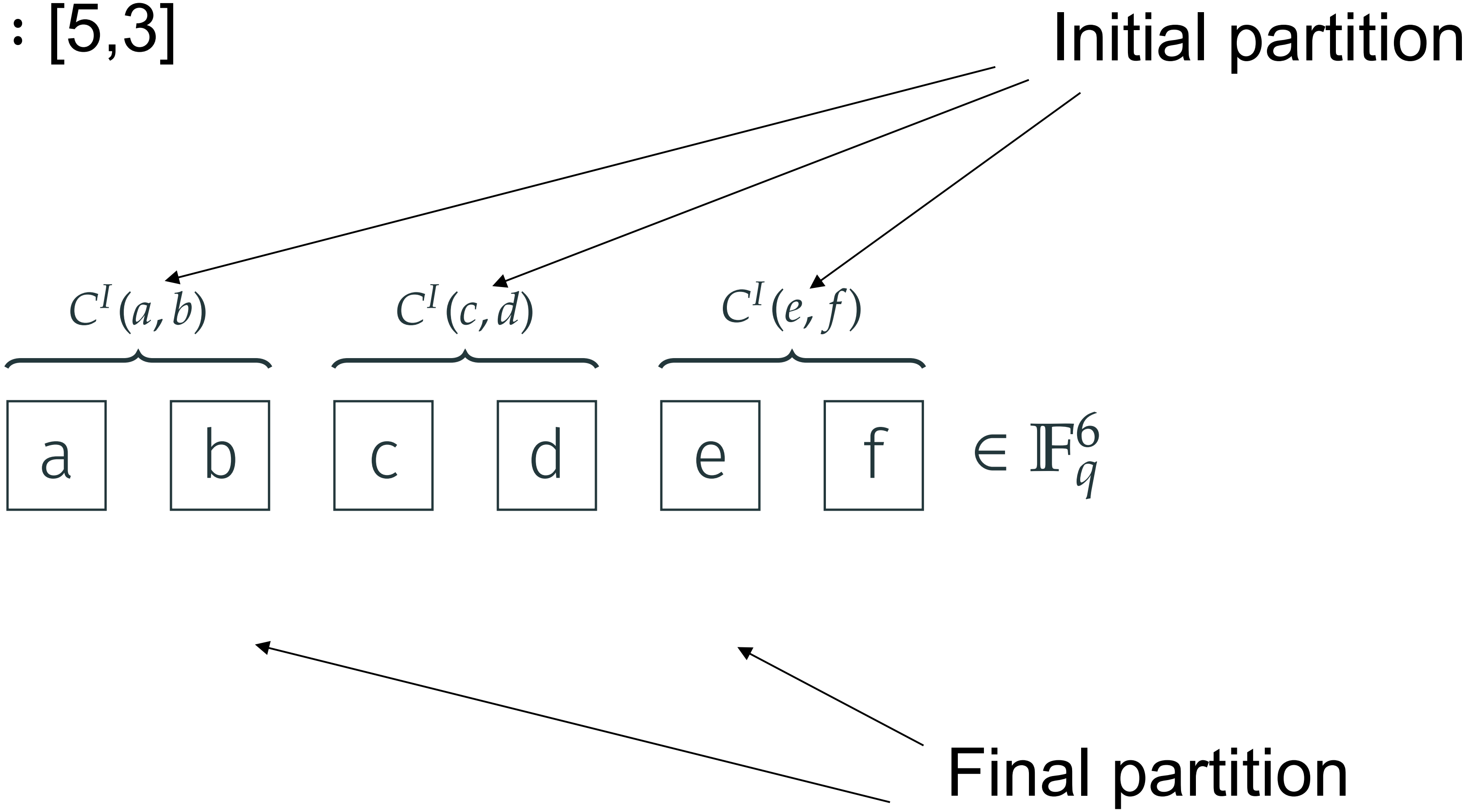
Example: Code conversion

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

a	b	c	d	e	f	$\in \mathbb{F}_q^6$

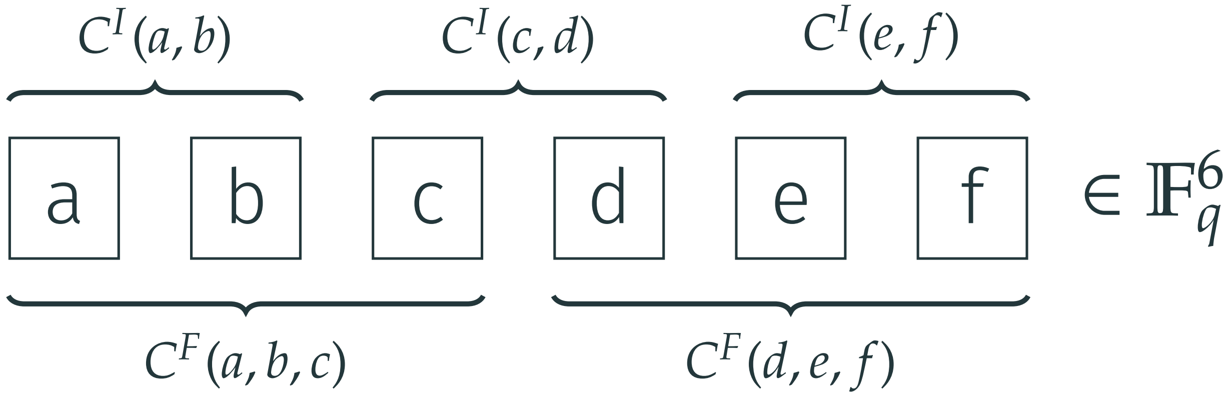
Example: Code conversion

- C^I : [3,2] ⇒ C^F : [5,3]



Example: Code conversion

- C^I : [3,2] ⇒ C^F : [5,3]



- For systematic codes

Convertible Codes

Definition [$(\boldsymbol{n^I}, \boldsymbol{k^I}; \boldsymbol{n^F}, \boldsymbol{k^F})$ 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 $\mathbb{F}_q$

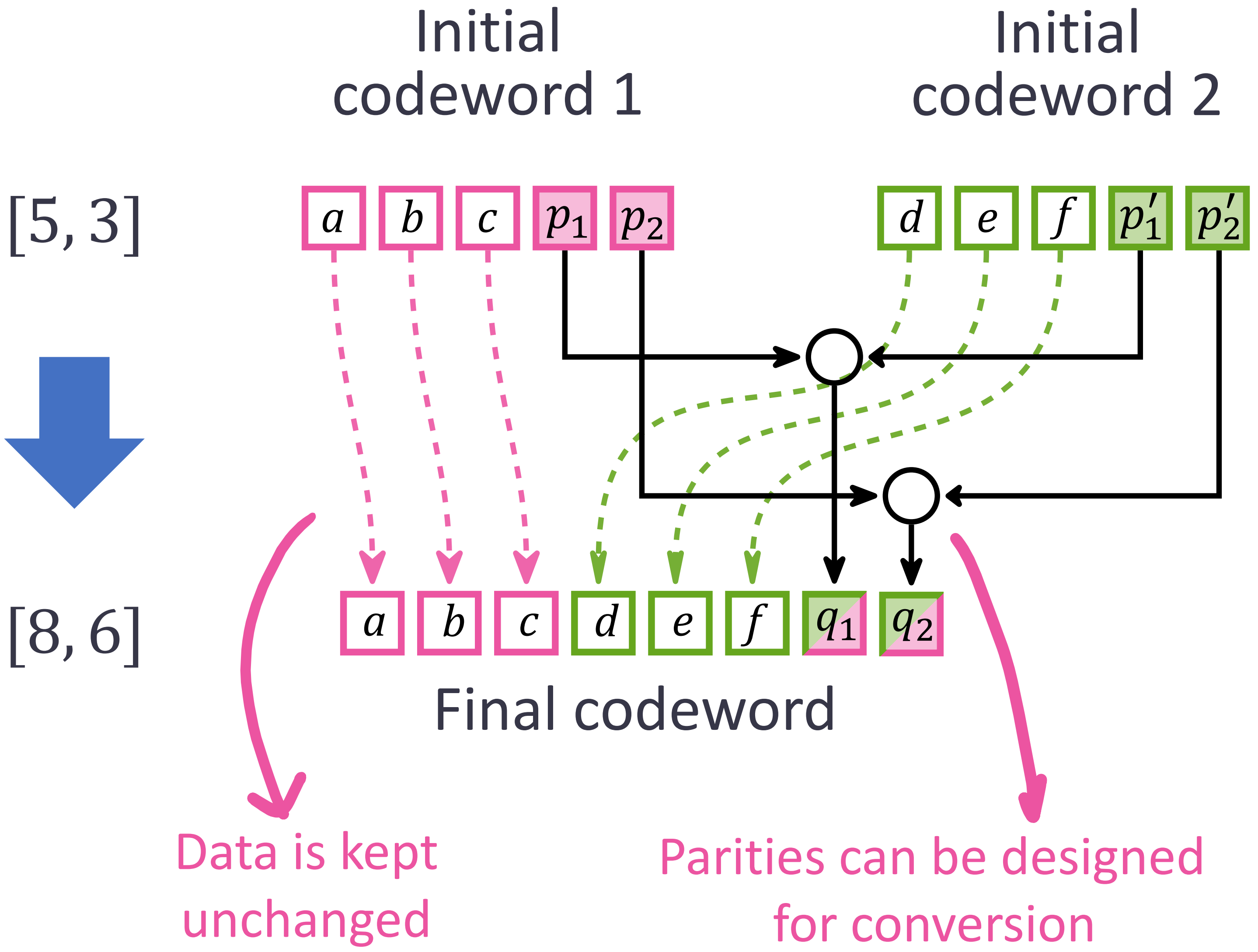
- C^I is an $[n^I, k^I]$ code; C^F is an $[n^F, k^F]$ code

(2) a pair of partitions $\mathcal{P}^I$, $\mathcal{P}^F$ of $[M = \mathrm{lcm}(k^I, k^F)]$

- 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 $\mathcal{T}_{C^I \to C^F}$

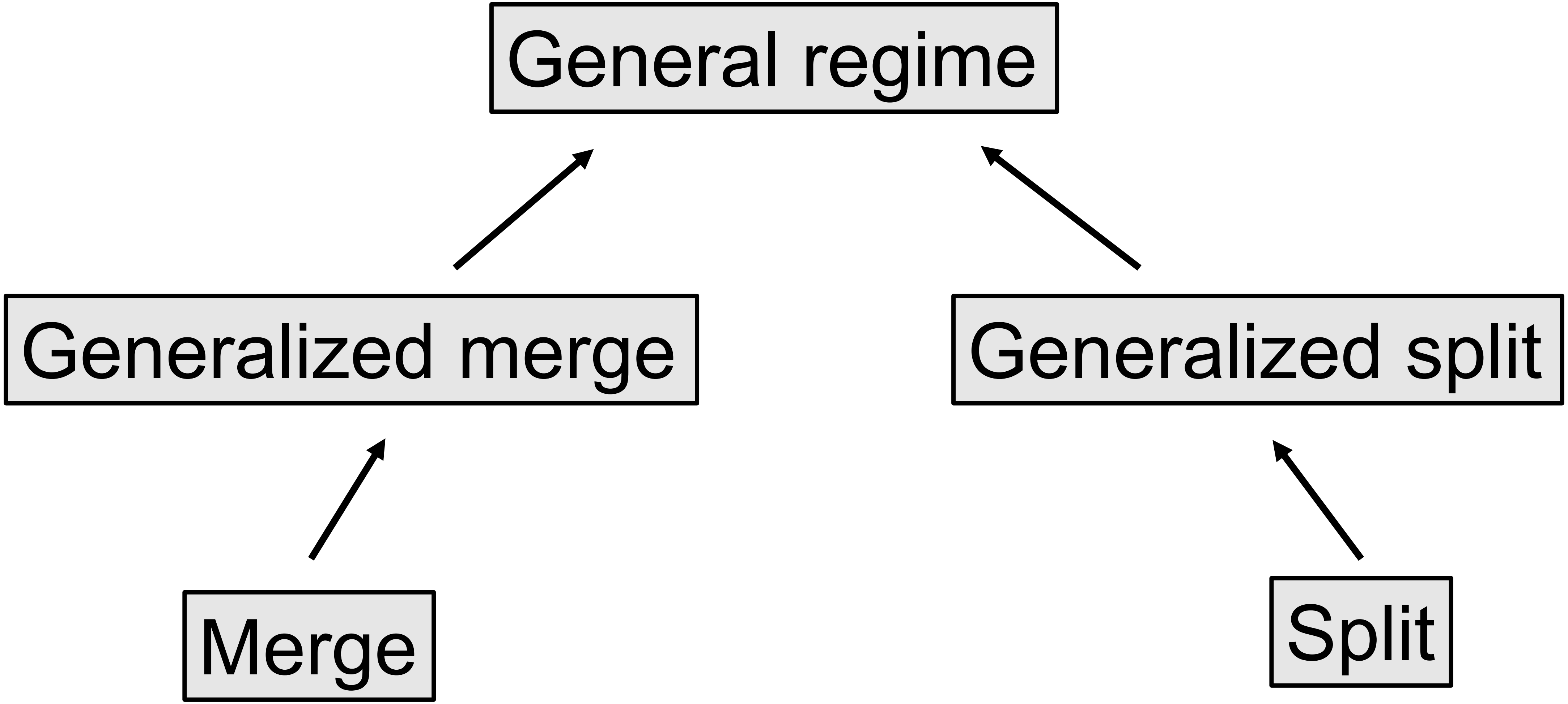
Code conversion: Toy example



$GF(7)$

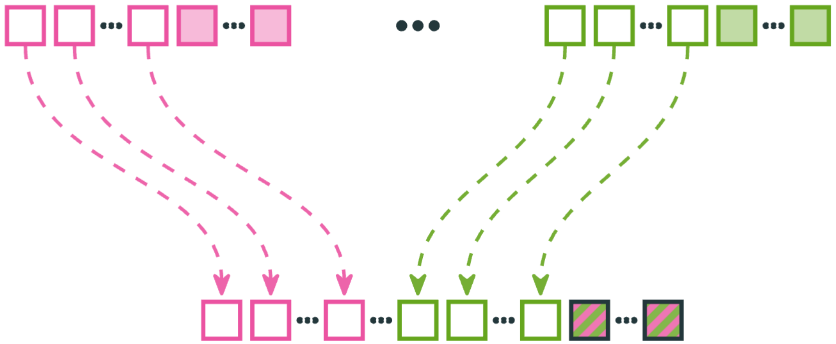
$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} d & e & f \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 6 \end{bmatrix} = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 2 \\ 1 & 6 \\ 1 & 4 \\ 1 & 5 \end{bmatrix}$$

Types of code conversions



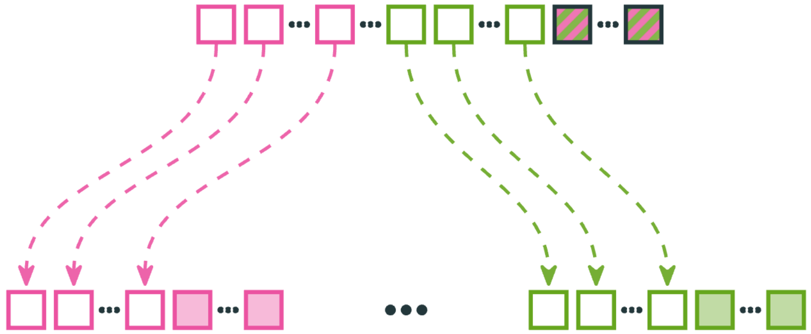
Code conversion regimes

Merge regime



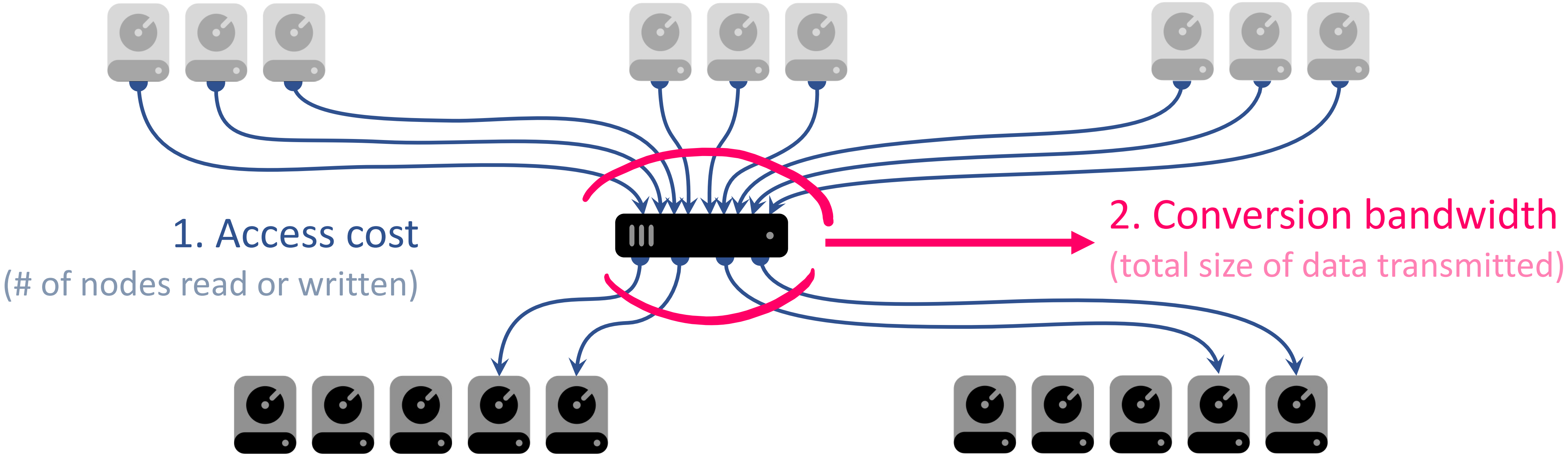
$$k^F = \lambda^I k^I$$

Split regime



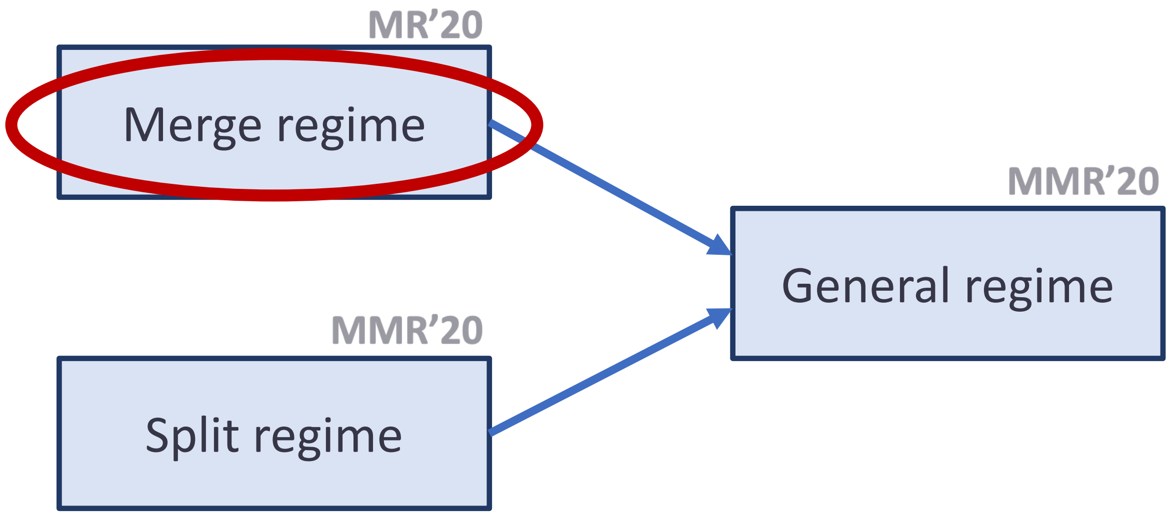
$$k^I = \lambda^F k^F$$

Cost of code conversion

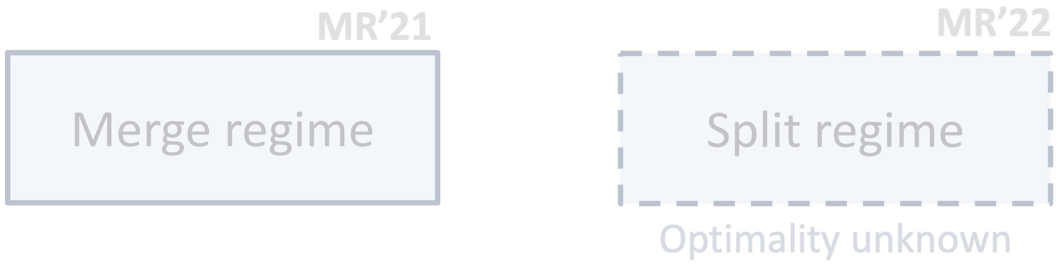


Cost of code conversion for linear MDS codes

Access cost



Conversion bandwidth



Tight lower bounds + explicit optimal constructions

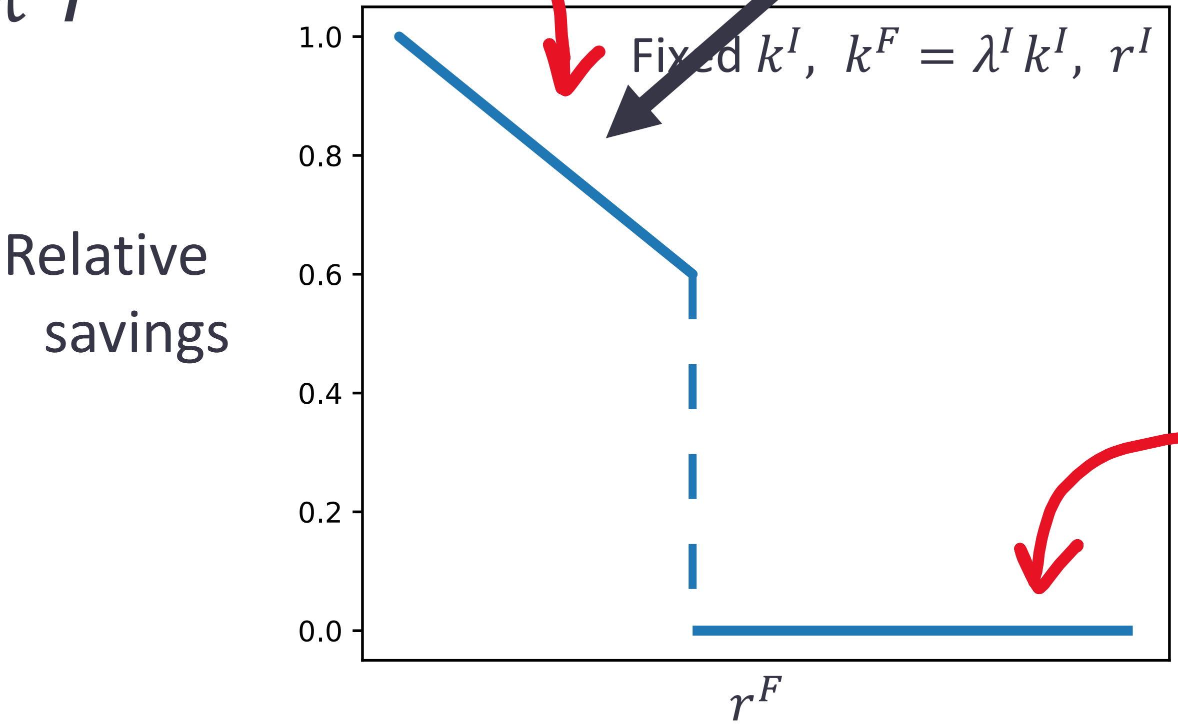
Access cost: lower bound

Theorem

(r = n – k = number of parities for a systematic code)

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

How to construct codes here?



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

Constructing codes

A	B	C	A +B +C	A +2B +3C

$$[A\ \ B\ \ C]\begin{pmatrix} 1 & & & 1 & 1 \\ & 1 & & 1 & 2 \\ & & 1 & 1 & 3 \end{pmatrix}$$

systematic

parity matrix

$\boldsymbol{P}$

Properties for efficient conversion

Super-regular + Block-constructible

= Optimal MDS convertible code

Super-regular

Parity matrix
$\boldsymbol{P}$

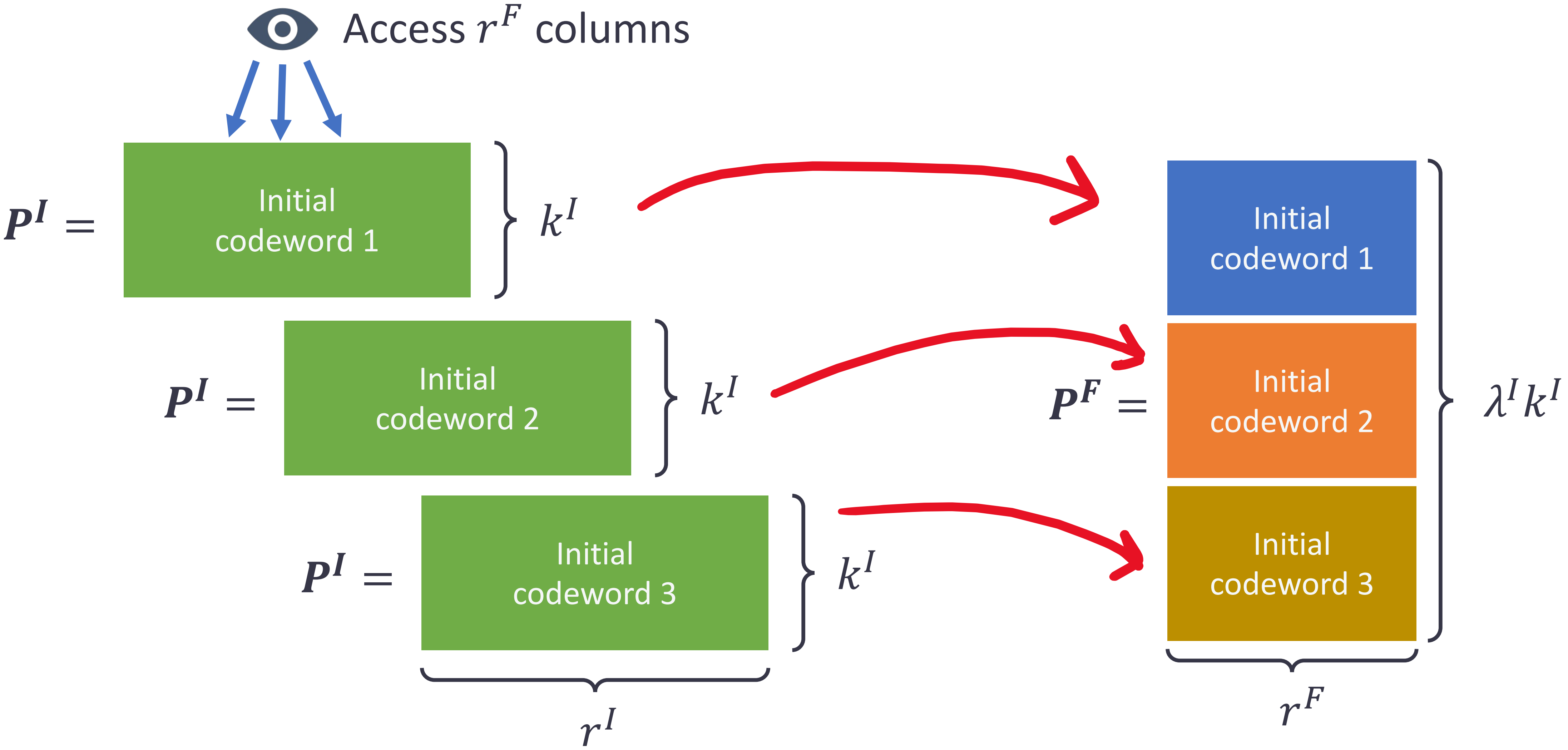
$$k \left\{ \overbrace{\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{pmatrix}}^{r} \right.$$

Elements from
Finite field $\mathbb{F}_q$

Systematic + MDS $\Leftrightarrow$ super-regular

Every square submatrix is invertible

Block-constructible



Each block of $\boldsymbol{P^F}$ is spanned by $\boldsymbol{r^F}$ columns of $\boldsymbol{P^I}$

Access cost: construction

Theorem (informal)

Construction of access-optimal convertible codes for all merge regime parameters for large enough field sizes.

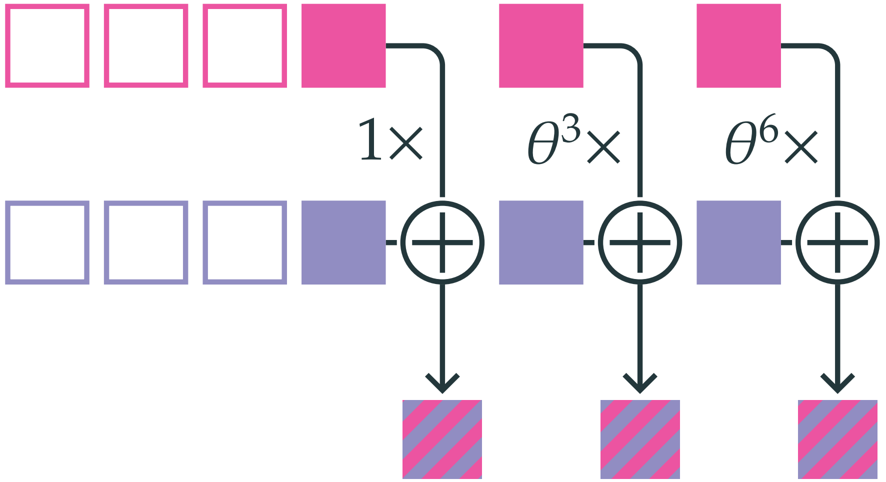
Access cost: general construction

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

θ: primitive element

$$\mathbf{P}^I = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \theta & \theta^2 \\ 1 & \theta^2 & \theta^4 \end{bmatrix}$$

$$\mathbf{P}^F = \left[\begin{array}{ccc} 1 & 1 & 1 \\ 1 & \theta & \theta^2 \\ 1 & \theta^2 & \theta^4 \\ \hline 1 & \theta^3 & \theta^6 \\ 1 & \theta^4 & \theta^8 \\ 1 & \theta^5 & \theta^{10} \end{array}\right]$$



Conversion process

- ☑ Block-constructible
- ☑ Super-regular

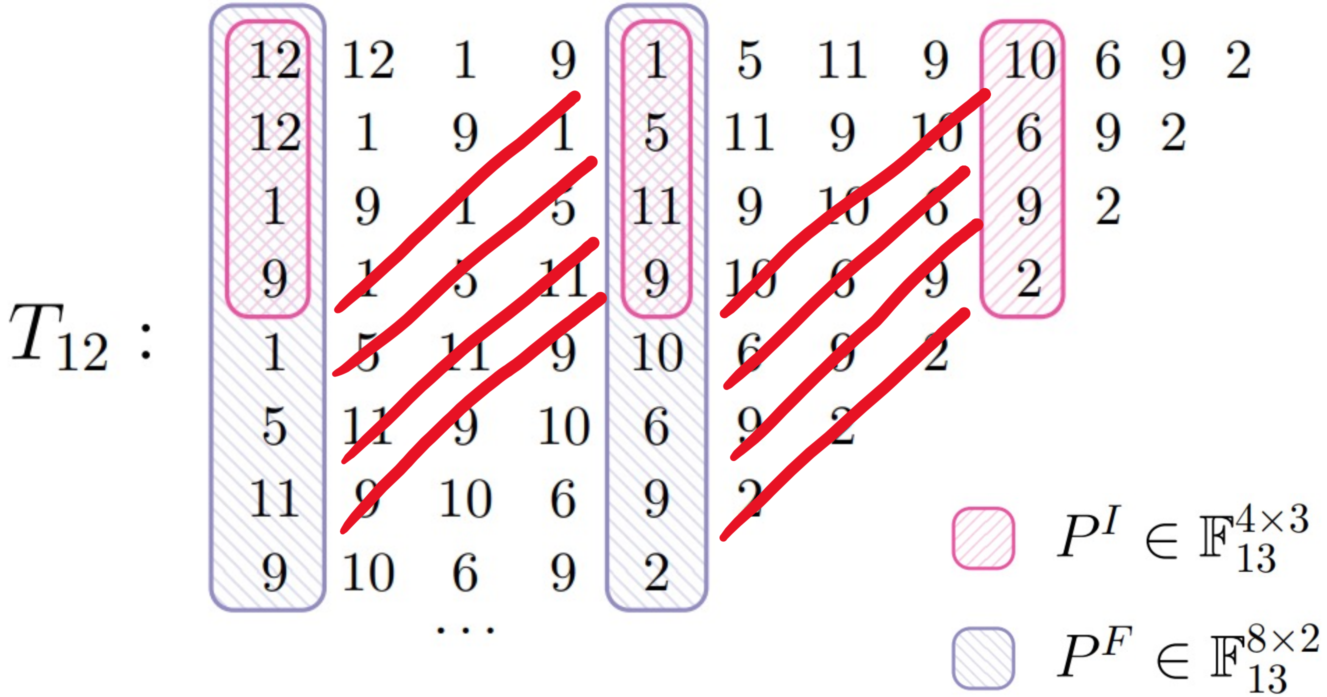
Requires high field size to ensure super-regularity

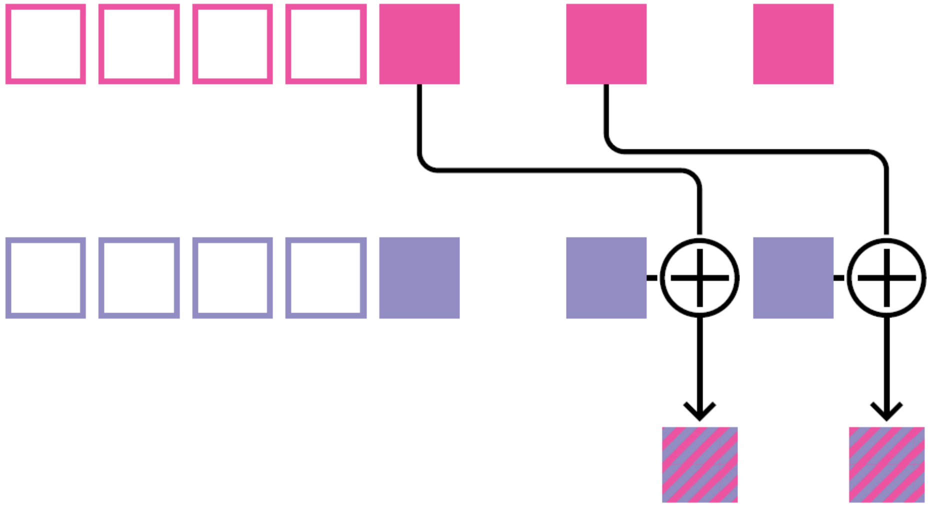
Access cost: construction

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





- ☑ Block-constructible
- ☑ Super-regular

Cost of code conversion for linear MDS codes

Access cost

MR'20

Merge regime

MMR'20

General regime

MMR'20

Split regime

Conversion bandwidth

MR'21

Merge regime

MR'22

Split regime

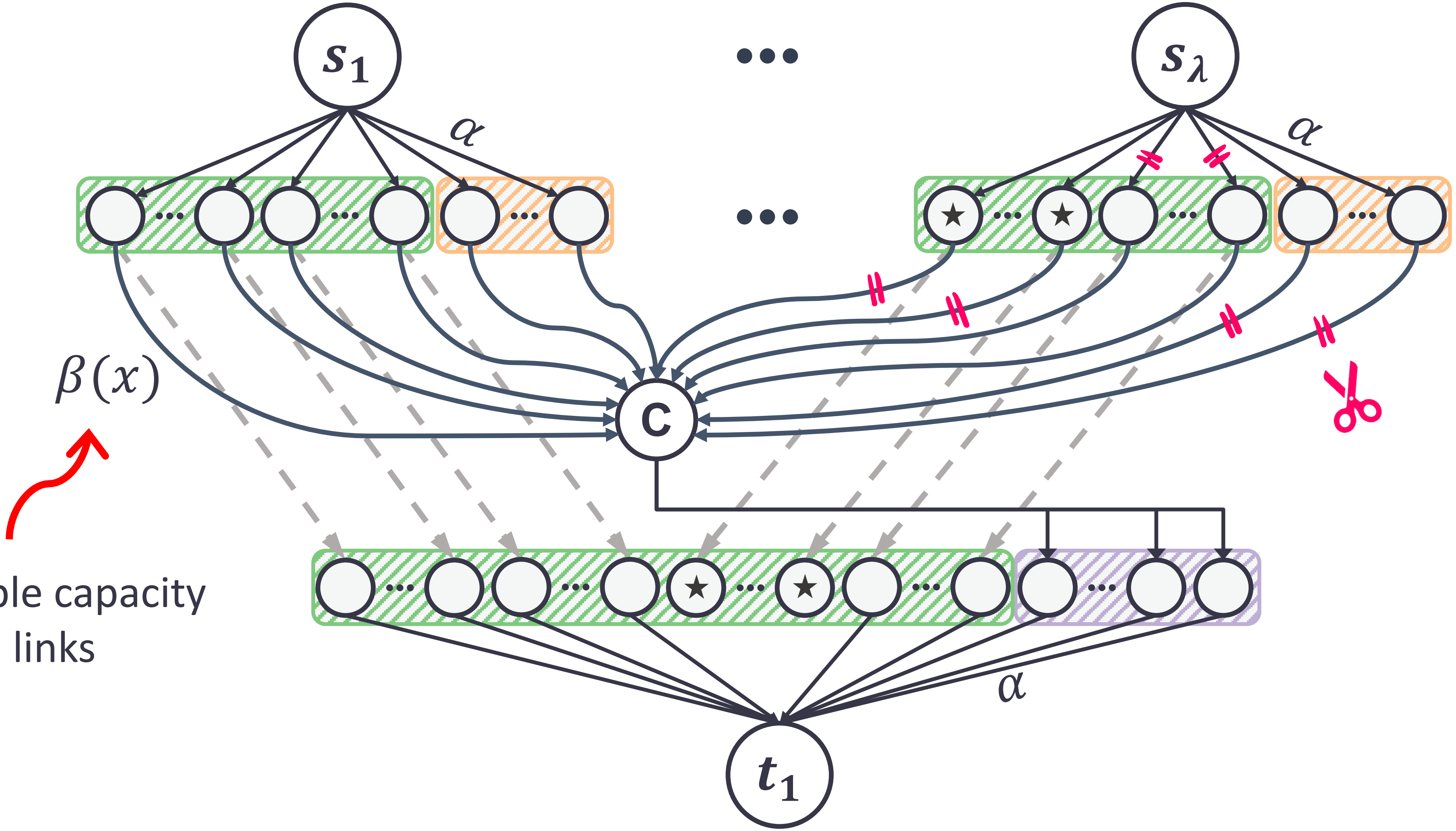
Optimality unknown

Tight lower bounds (for merge only) + explicit constructions

Conversion bandwidth

- Lower access cost already gives lower conversion bandwidth as well
- Can we achieve further reduction in conversion bandwidth over access-optimal convertible codes?

Bandwidth: lower bound



Variable capacity links

Bandwidth: lower bound

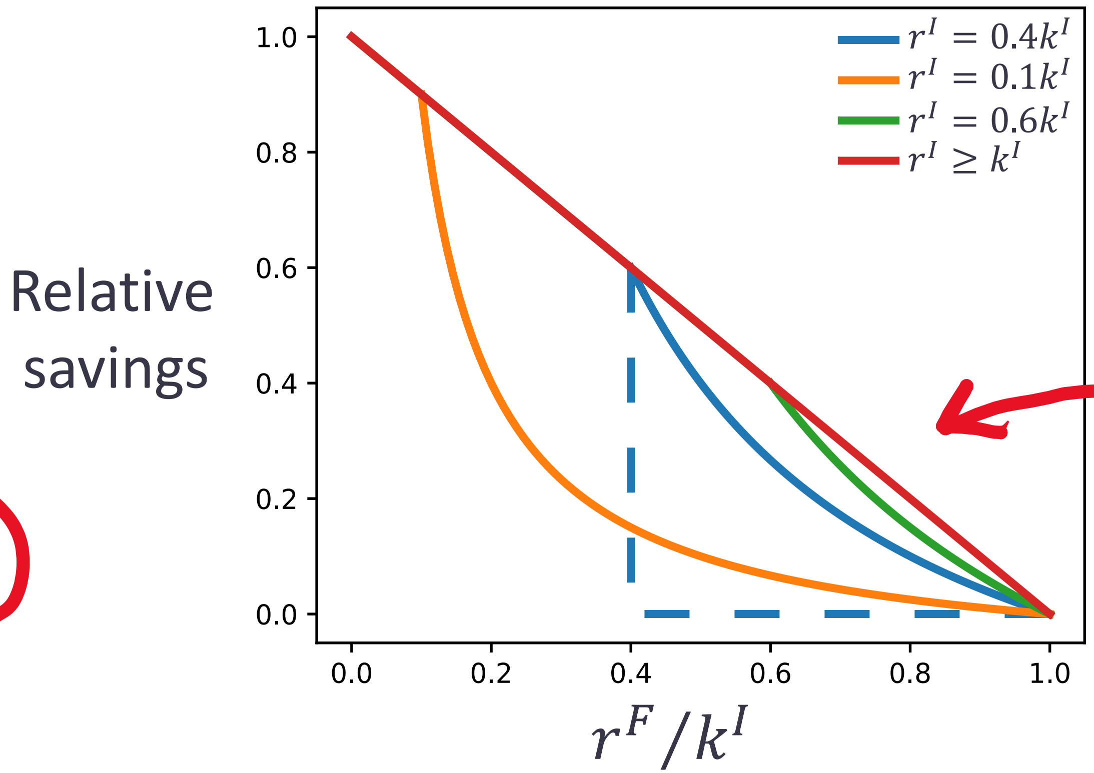
α: vector size

Theorem

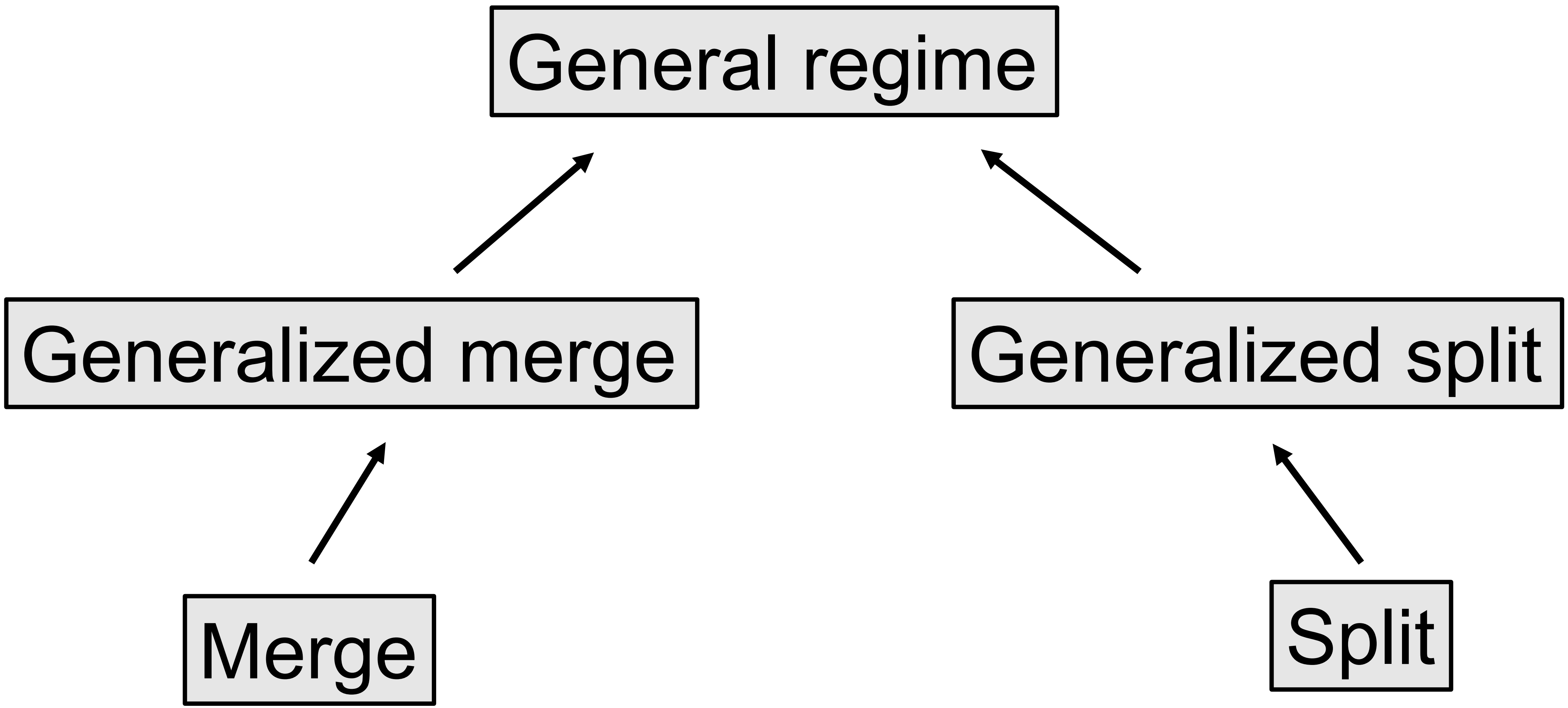
If $r^F \leq r^I$, BW cost is $\lambda^I r^F \alpha$

Same as access cost!

If $r^F > r^I$, BW cost is $\lambda^I k^I \alpha - \lambda^I r^I \alpha \left(\frac{k^I}{r^F} - 1\right)$

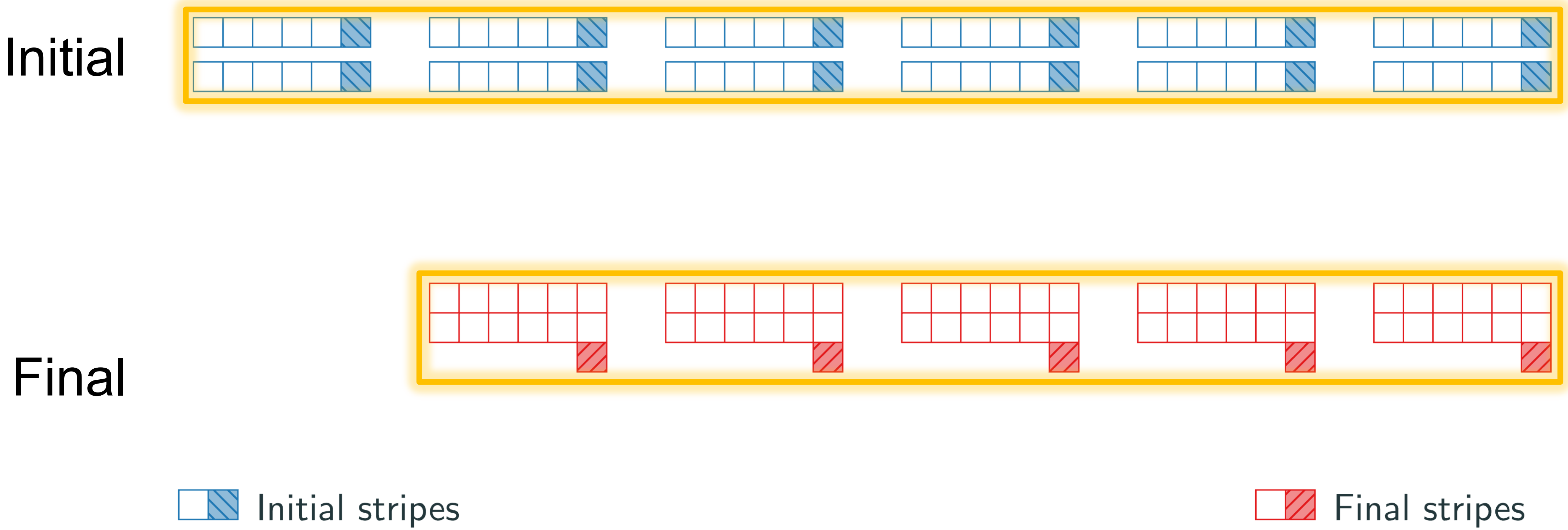


Building blocks of code conversion



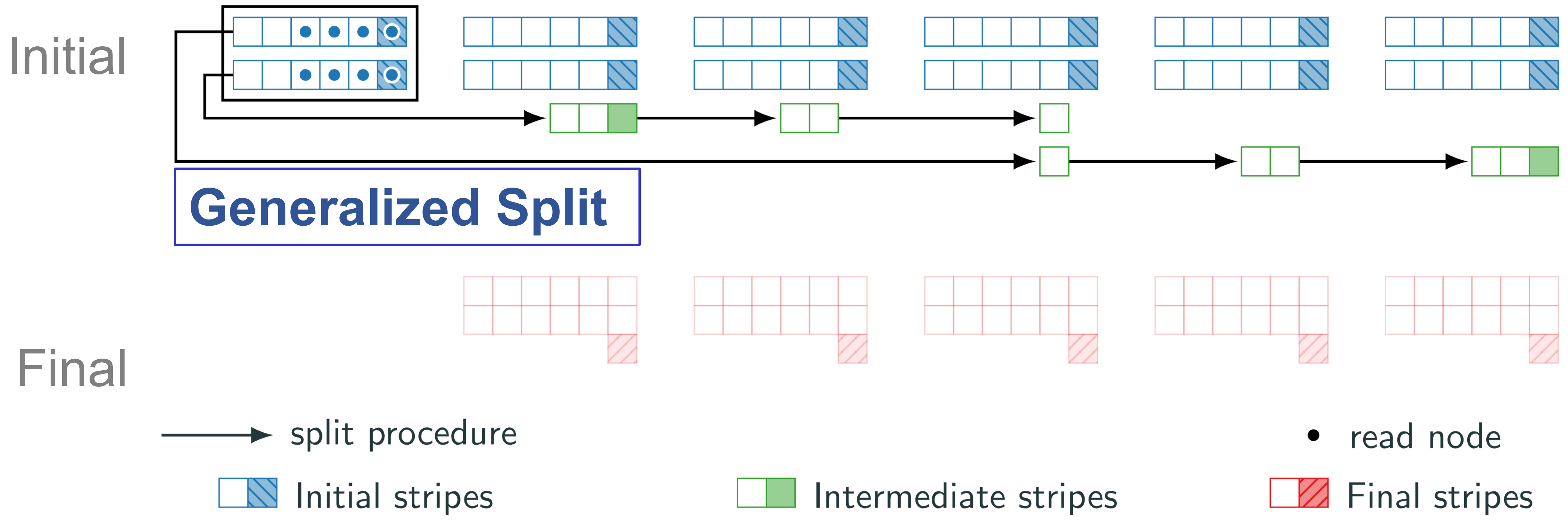
General conversion

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



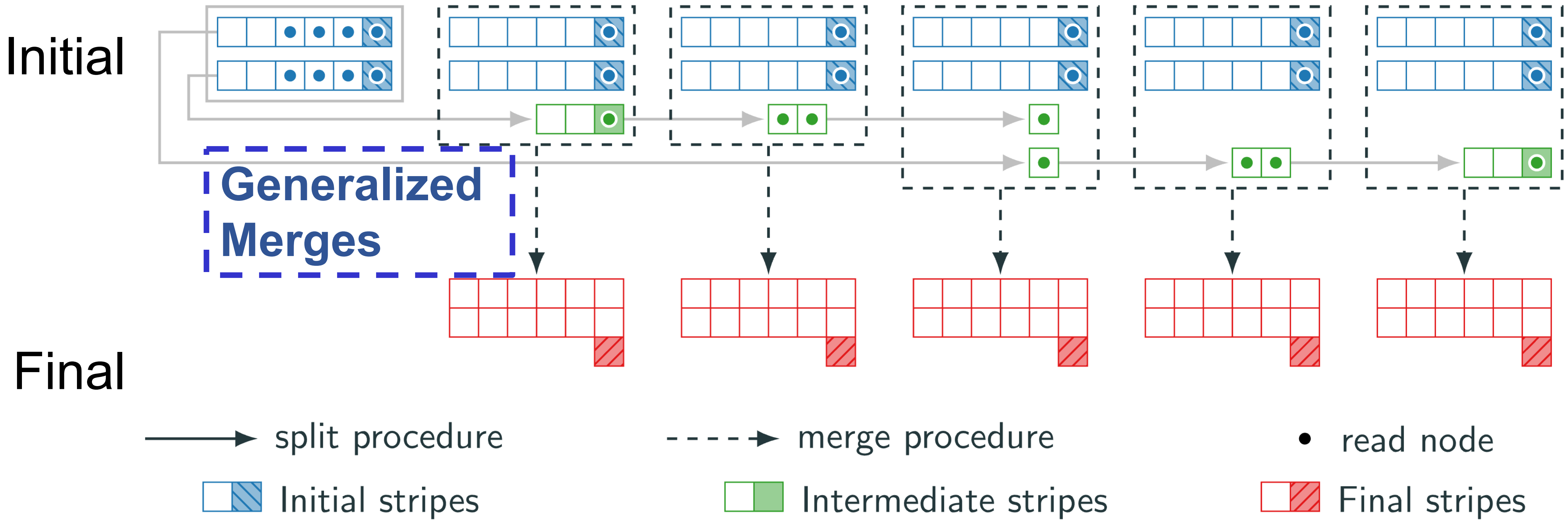
General conversion

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



General conversion

- 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?