18-847F: Special Topics in Computer Systems

Foundations of Cloud and Machine Learning Infrastructure

Lecture 9: Coding for Distributed Storage

Foundations of Cloud and Machine Learning Infrastructure

Outline

Coded Distributed Storage

Repair-efficiency

Service Capacity

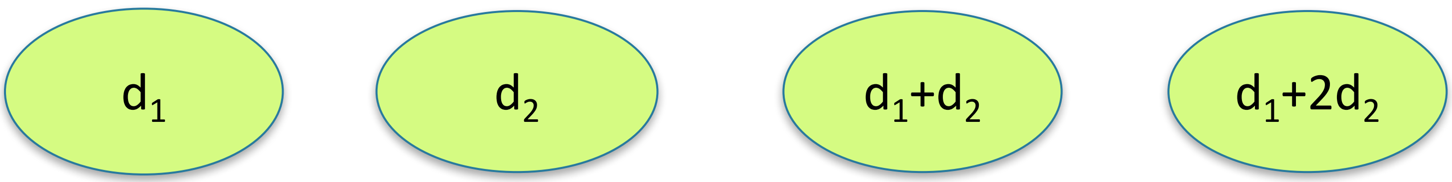
(n,k) Reed-Solomon Codes: 1960

- Data: $d_1, d_2, d_3, \ldots d_k$
- Polynomial: $d_1 + d_2\, x + d_3\, x^2 + \ldots d_k\, x^{k-1}$
- Parity bits: Evaluate at n-k points:

x=1:	$d_1 + d_2 + d_3 + d_4$
x=2:	$d_1 + 2\, d_2 + 4\, d_3 + 8\, d_4$
x=3 :	….
x=4:	….
x=n:	…

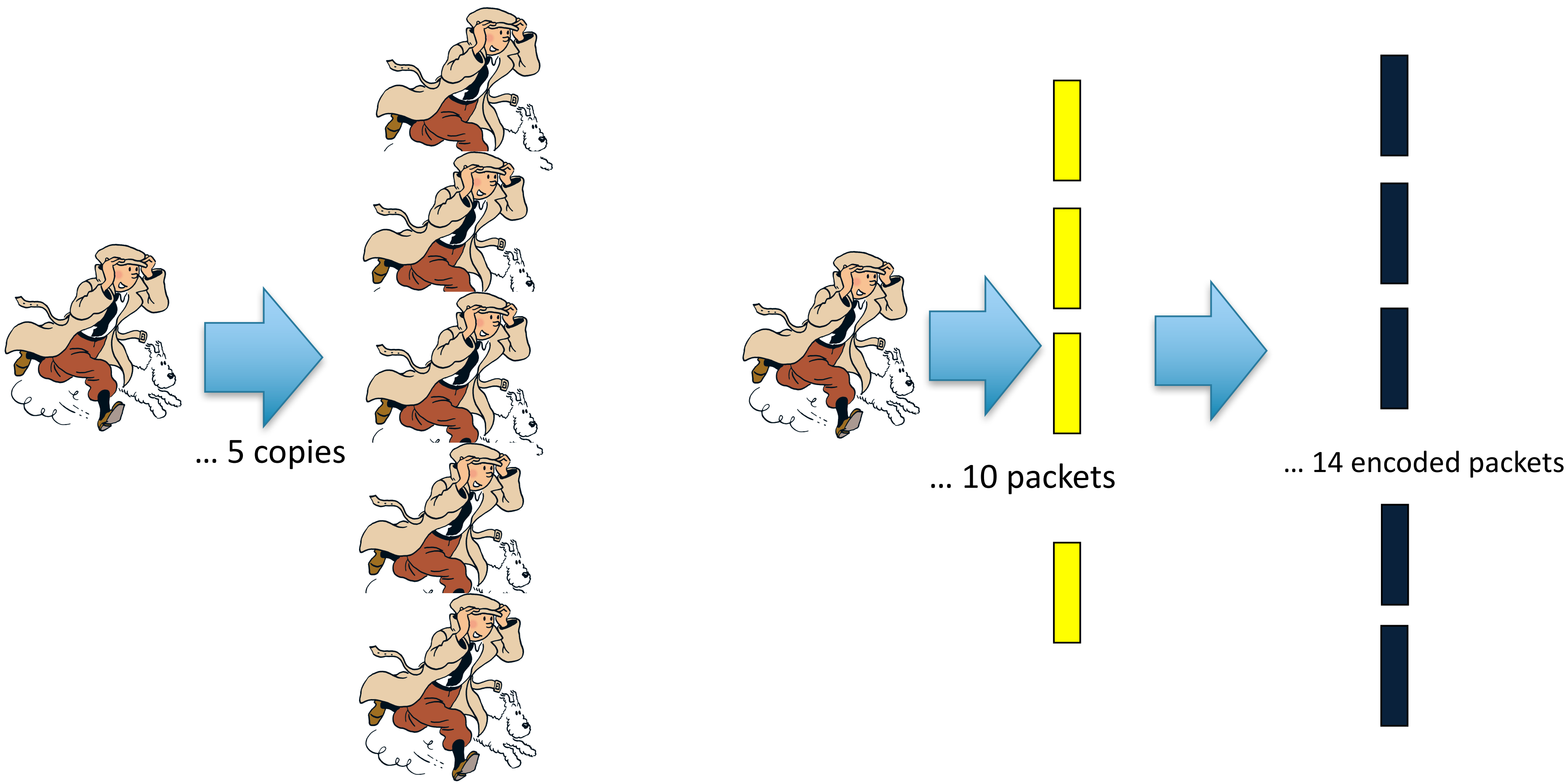
- Can solve for the coefficients from any k coded symbols

Example: (4,2) Reed-Solomon Code

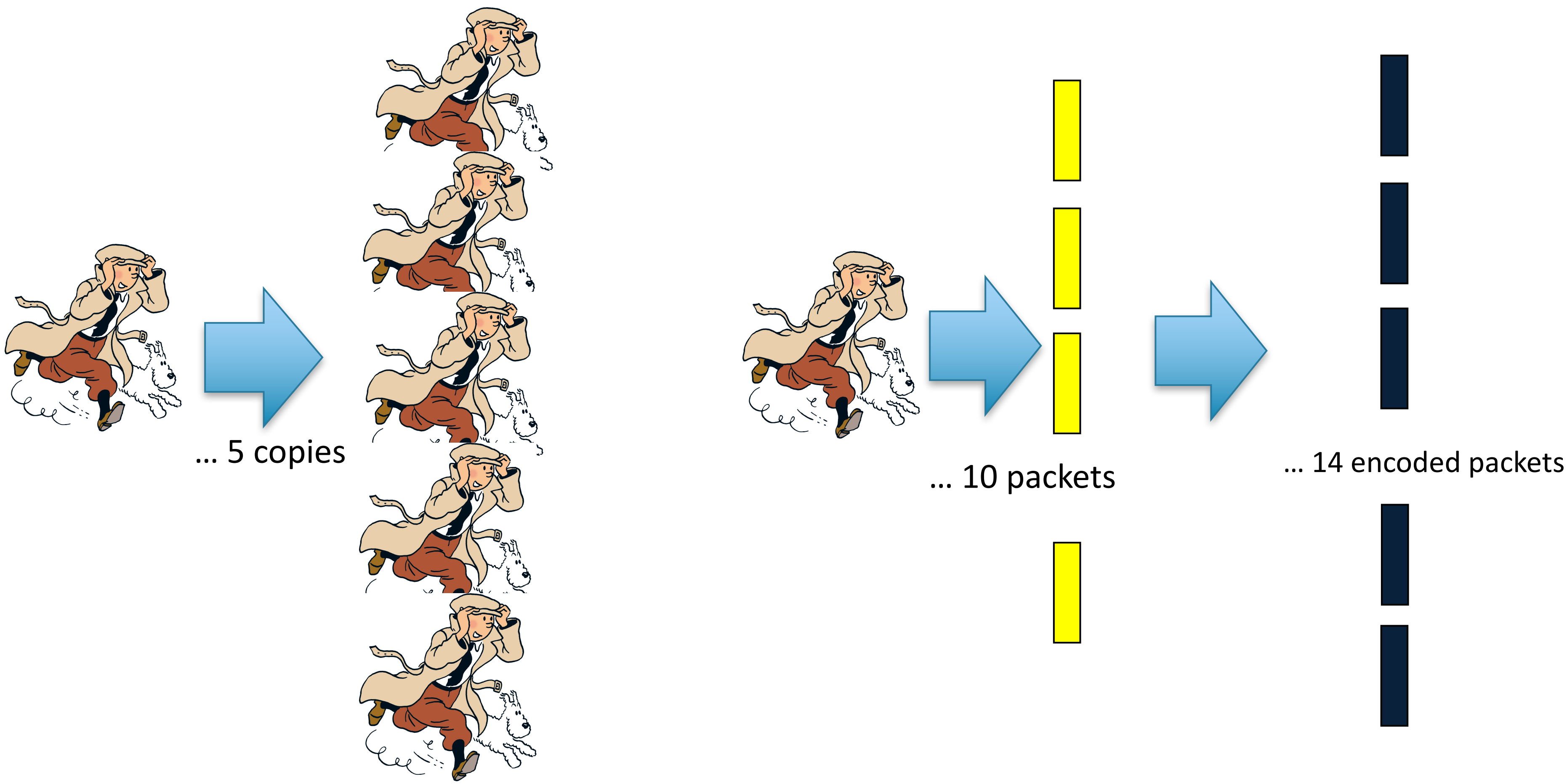


- Can solve for the coefficients from any k coded symbols
- Microsoft uses (7, 4) code
- Facebook uses (14,10) code

Coding vs Replication

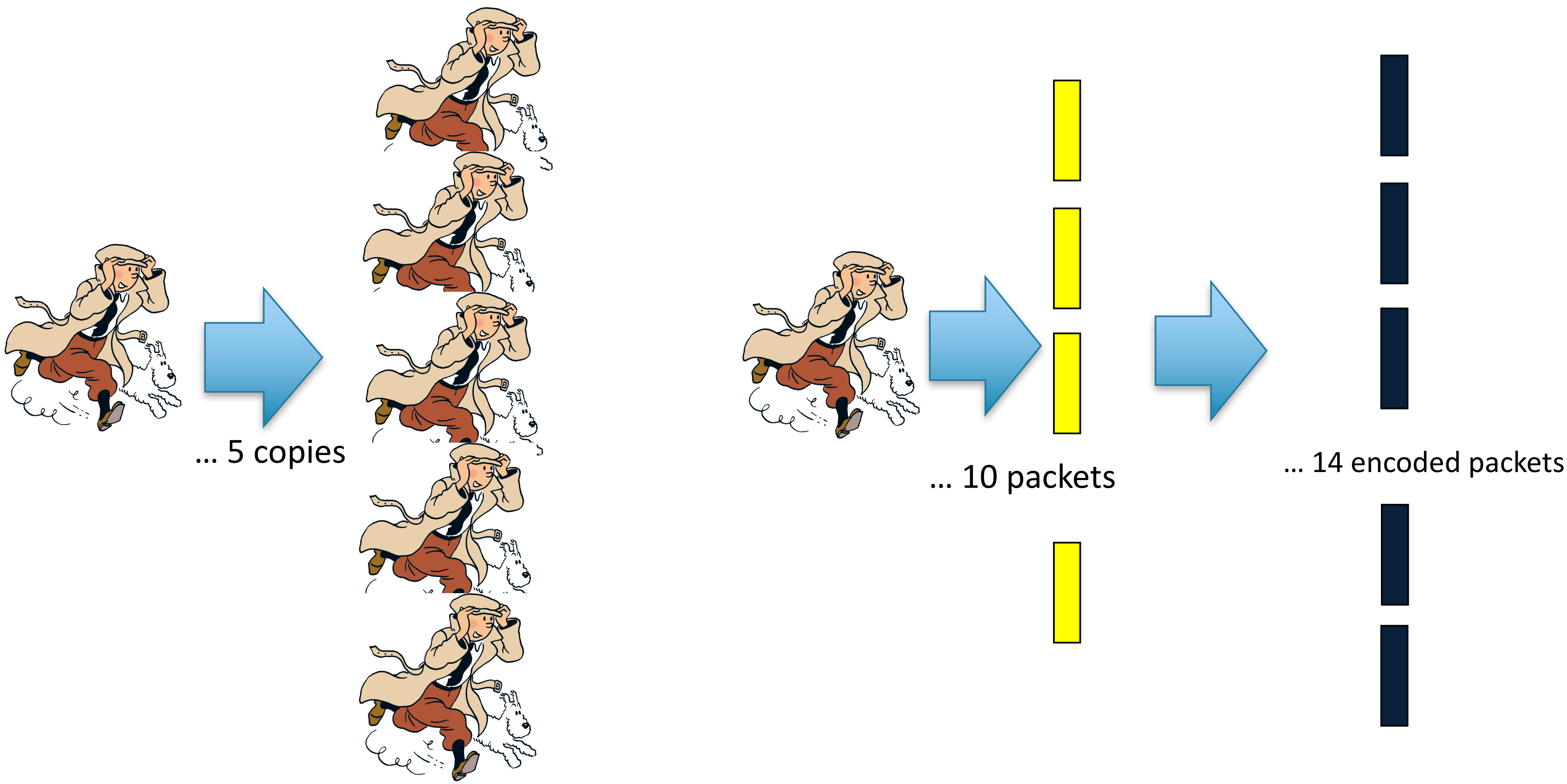


Concept Check: Coding vs Replication



- How many node-failures can each system tolerate?
- What is the code rate of each system?

Concept Check: Coding vs Replication



- How many node-failures can each system tolerate?: 4
- What is the code rate of each system? 1/5 and 10/14
- Replication uses 357% more storage for the same reliability!

Outline

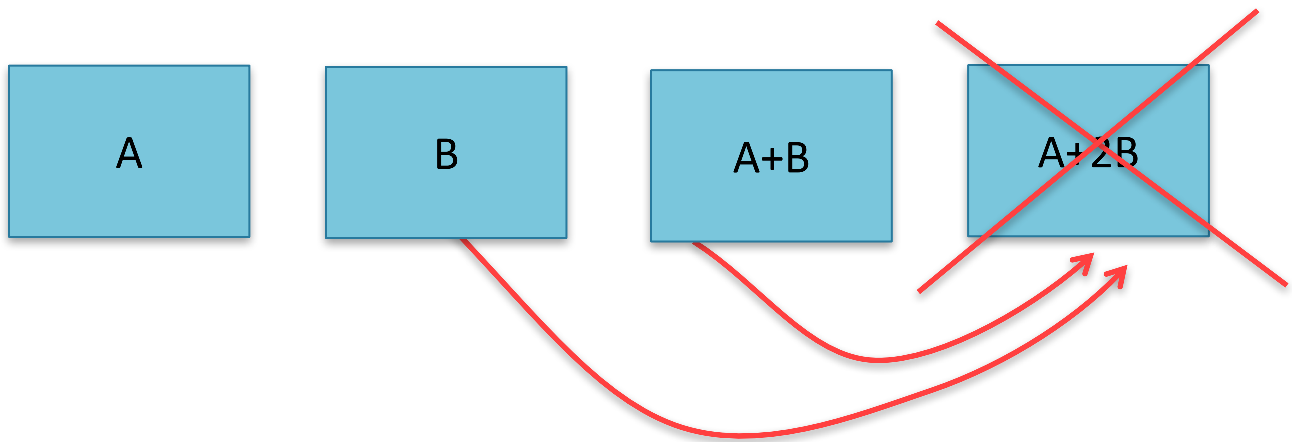
Coded Distributed Storage

Repair-efficiency

Service Capacity

Locality and Repair Issues

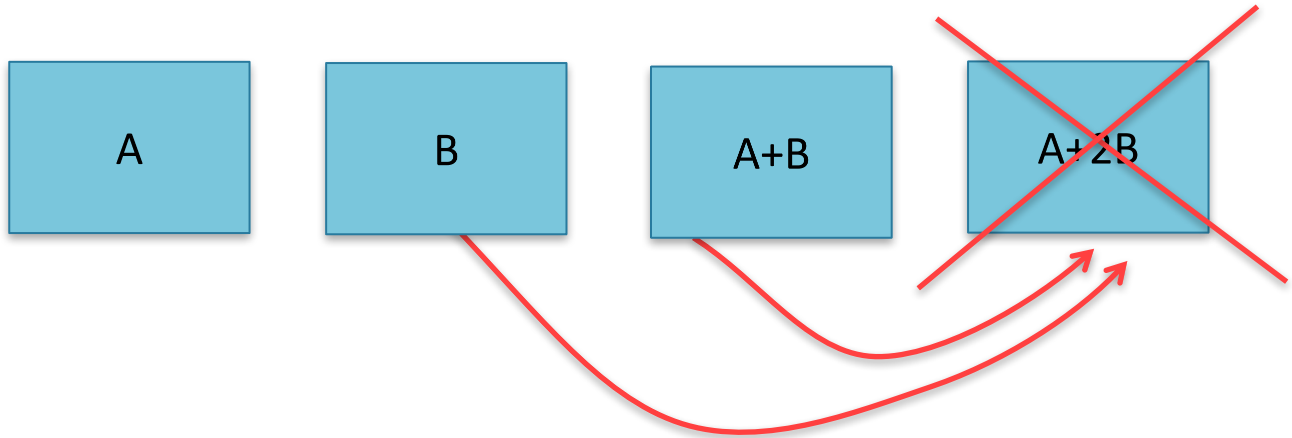
- Most distributed storage systems still use replication (21x in Gmail!)
- Repairing failed nodes is hard with Reed-Solomon Codes..



- If we lose 1 node :
  - Need to contact k other nodes
  - Need to download k times the lost data

Locality and Repair Issues

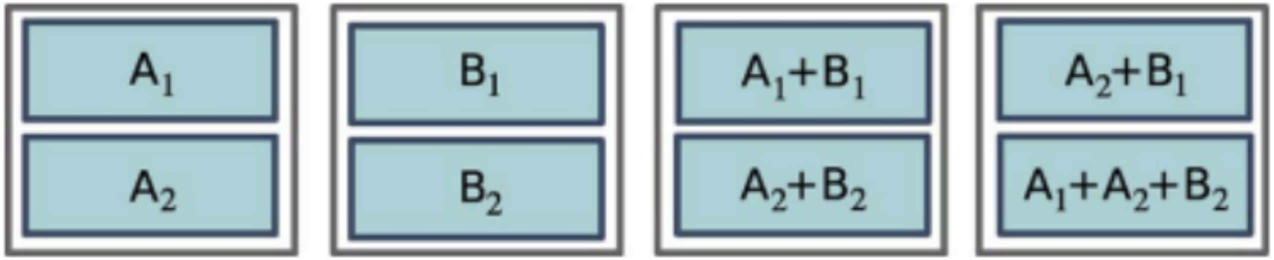
- Most distributed storage systems still use replication (21x in Gmail!)
- Repairing failed nodes is hard with Reed-Solomon Codes..



- If we lose 1 node:
  - Need to contact k other nodes
  - Need to download k times the lost data

Locality and Repair Issues

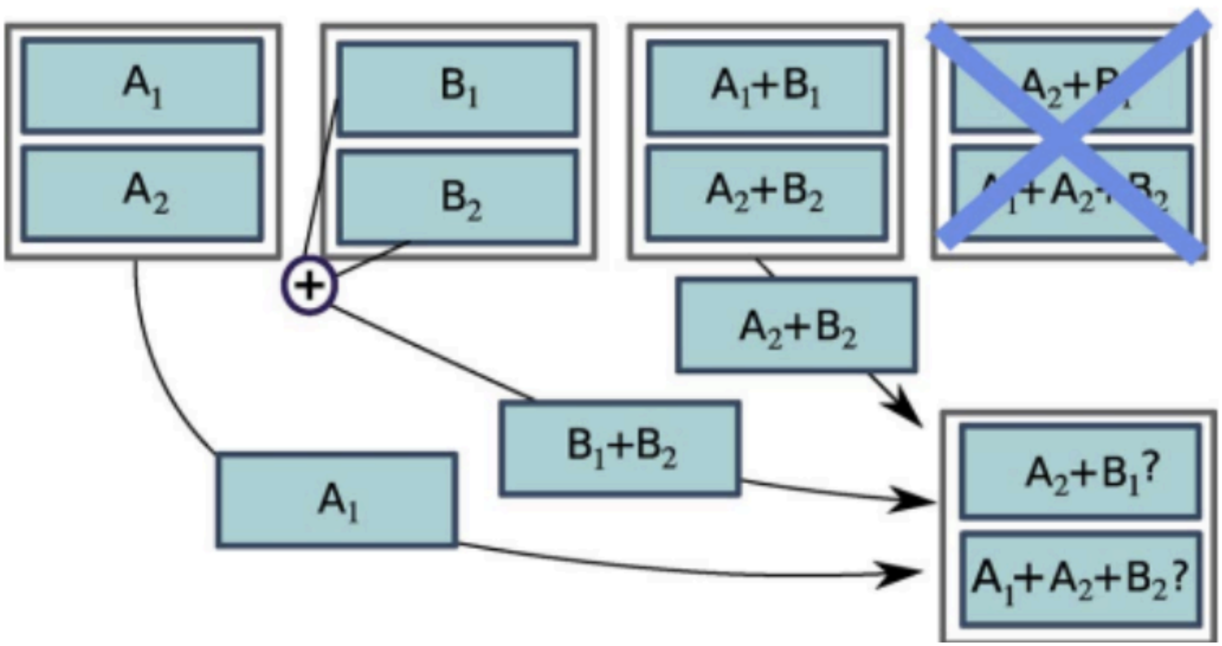
- Most distributed storage systems still use replication (21x in Gmail!)
- Repairing failed nodes is hard with Reed-Solomon Codes..



- If we lose 1 node:
  - Need to contact k other nodes
  - Need to download k times the lost data

Solution: Regenerating Codes

- Codes designed to minimize:
  - Repair Bandwidth
  - Number of nodes contacted



Exact vs Functional Repair

Exact repair

Repair the failed nodes exactly

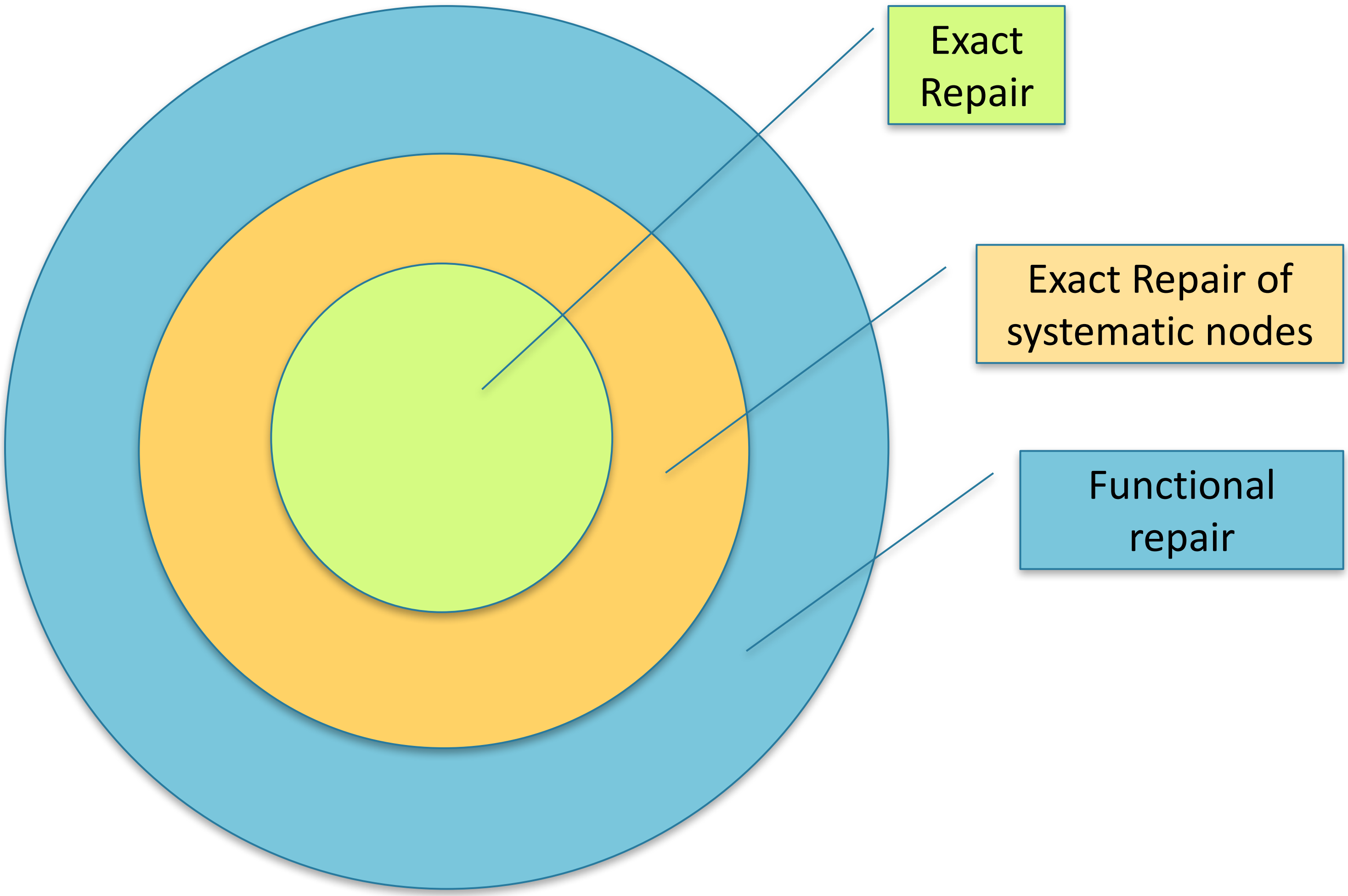
Functional repair

New data should be equivalent to the old for repair purposes, that is, k out of n nodes are still enough for repair

Exact repair of systematic nodes

Systematic nodes should be repaired exactly. Other notes may be repaired functionally

Exact vs Functional Repair



Model 1: Functional Repair

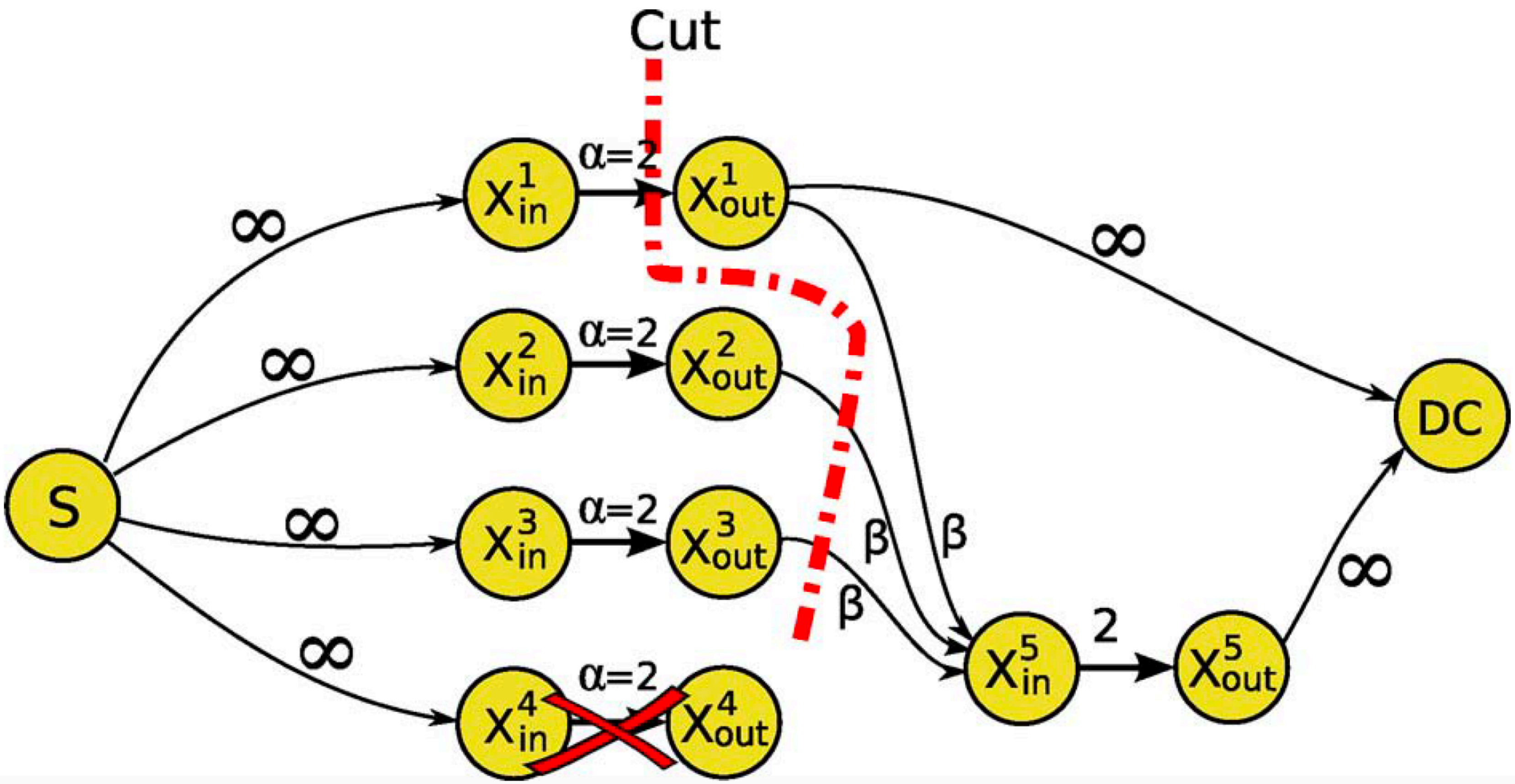
- File of size M, stored on n nodes, with α bits per node
- A failed node can be repaired using any d surviving nodes
- Each of the d nodes send β bits to repair it
- Repair bandwidth = γ = $d\beta$

[Dimakis et al 2008] studies the fundamental trade-off b/w

Storage per node: α and

Repair bandwidth: γ

Information flow graph model



The min-cut needs to larger than M in order to recover the file

[Dimakis et al 2008]:

Theorem 1: For any $\alpha \geq \alpha^*(n,k,d,\gamma)$, the points (n,k,d,α,γ) are feasible, and linear network codes suffice to achieve them. It is information theoretically impossible to achieve points with $\alpha < \alpha^*(n,k,d,\gamma)$. The threshold function $\alpha^*(n,k,d,\gamma)$ is the following:

$$\alpha^*(n,k,d,\gamma) = \begin{cases} \frac{\mathcal{M}}{k}, & \gamma \in [f(0), +\infty) \\ \frac{\mathcal{M}-g(i)\gamma}{k-i}, & \gamma \in [f(i), f(i-1)), \end{cases} \quad (1)$$

where

$$f(i) \triangleq \frac{2\mathcal{M}d}{(2k-i-1)i + 2k(d-k+1)}, \quad (2)$$

$$g(i) \triangleq \frac{(2d-2k+i+1)i}{2d}, \quad (3)$$

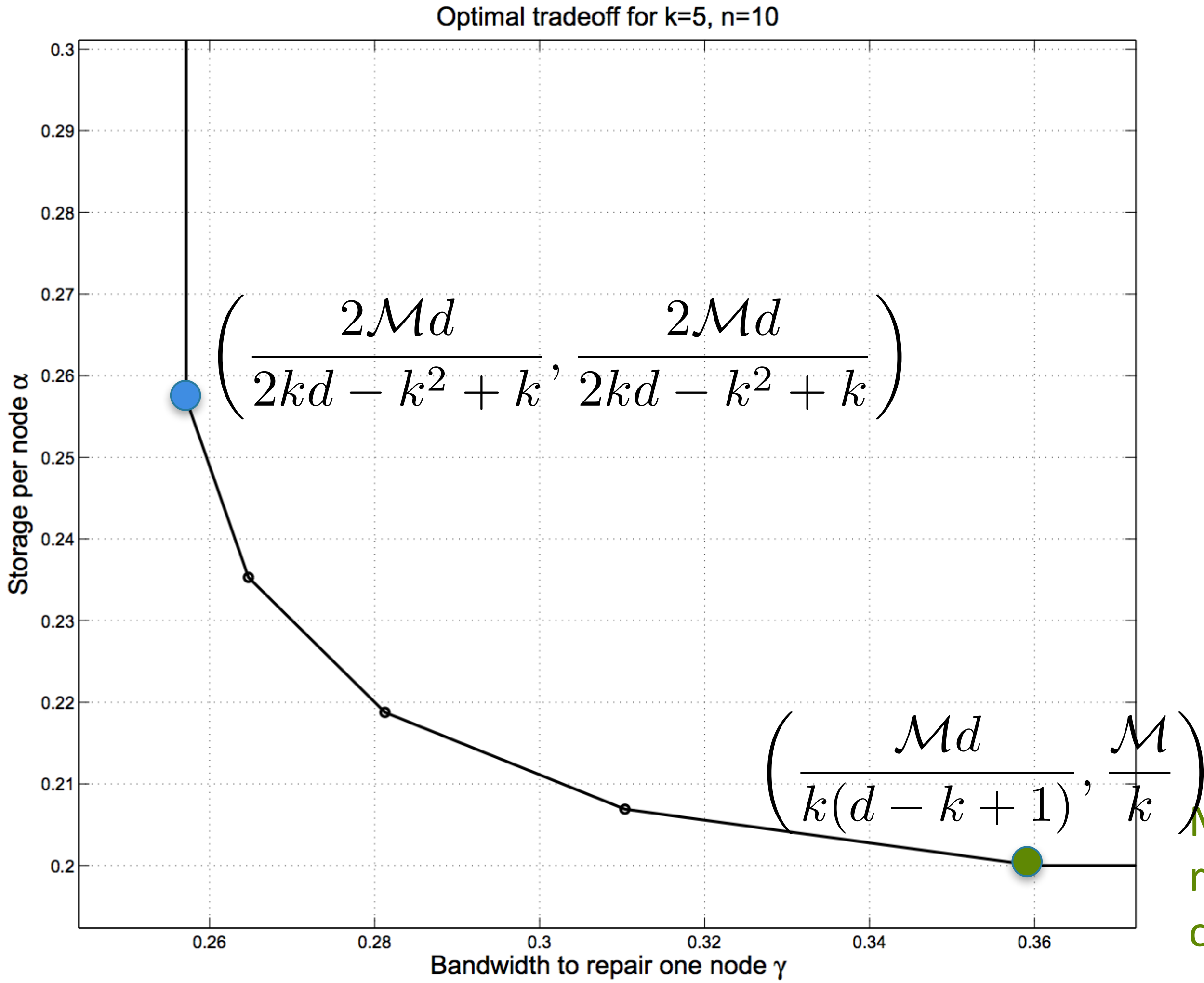
where $d \leq n-1$. For d, n, k given, the minimum repair bandwidth γ is

Decreases with d, minimum at d = n-1

$$\gamma_{\min} = f(k-1) = \frac{2\mathcal{M}d}{2kd - k^2 + k}. \quad (4)$$

Storage-Bandwidth Trade-off

Minimum b/w regenerating (MBR) codes



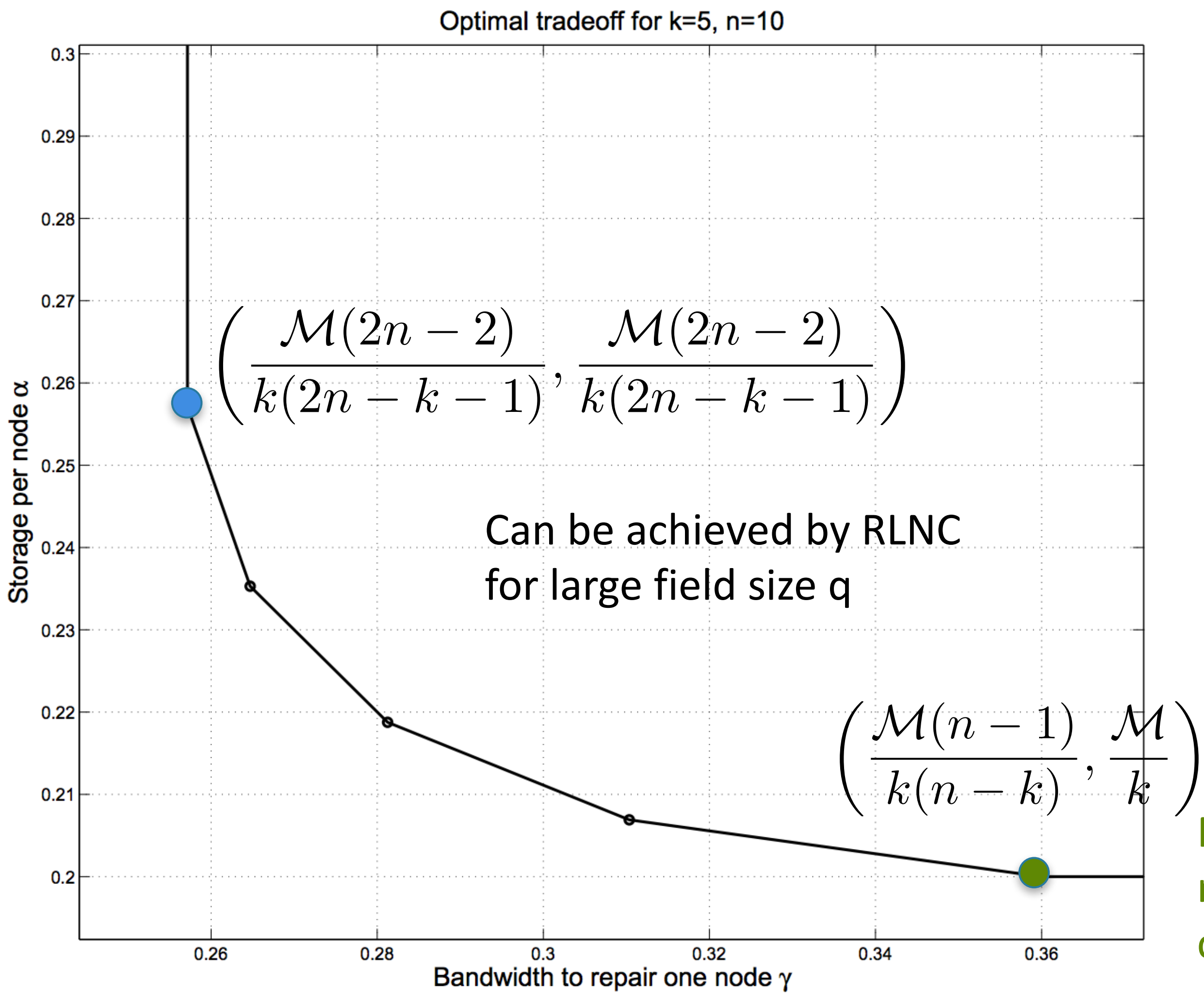
$$\left(\frac{2\mathcal{M}d}{2kd-k^2+k}, \frac{2\mathcal{M}d}{2kd-k^2+k}\right)$$

$$\left(\frac{\mathcal{M}d}{k(d-k+1)}, \frac{\mathcal{M}}{k}\right)$$

Minimum storage regenerating (MS codes

Storage-Bandwidth Trade-off

Minimum b/w regenerating (MBR) codes



Minimum storage regenerating (MSR) codes

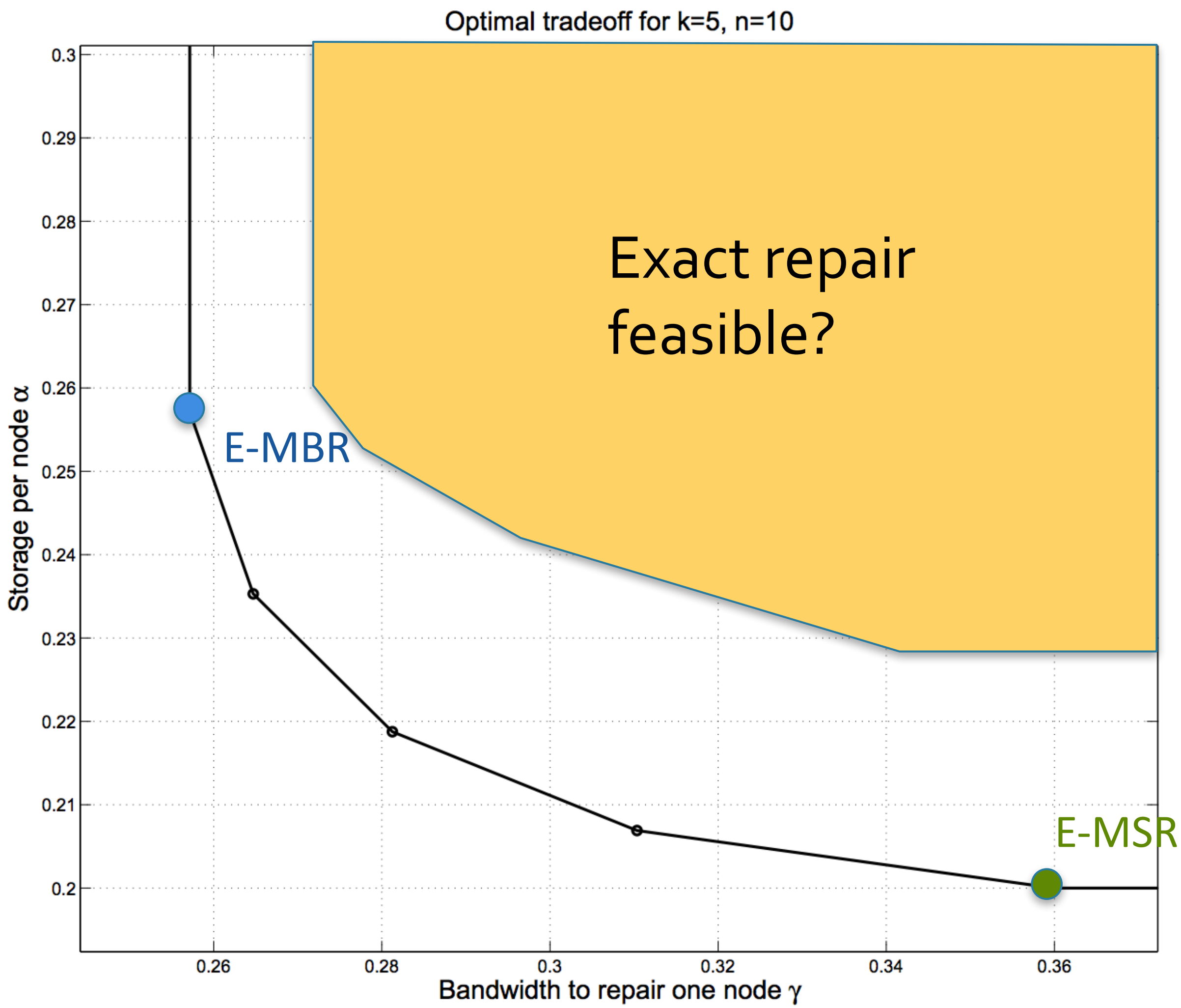
Concept Check: Min. Repair Bandwidth

Consider a file of size 1 Mb stored using an (7,4) code.

1. What is the repair-bandwidth of an (7,4) MDS code? How much data is stored at each node?

2. What is the min. possible repair bandwidth, for the same storage per node?

Model 2: Exact Repair

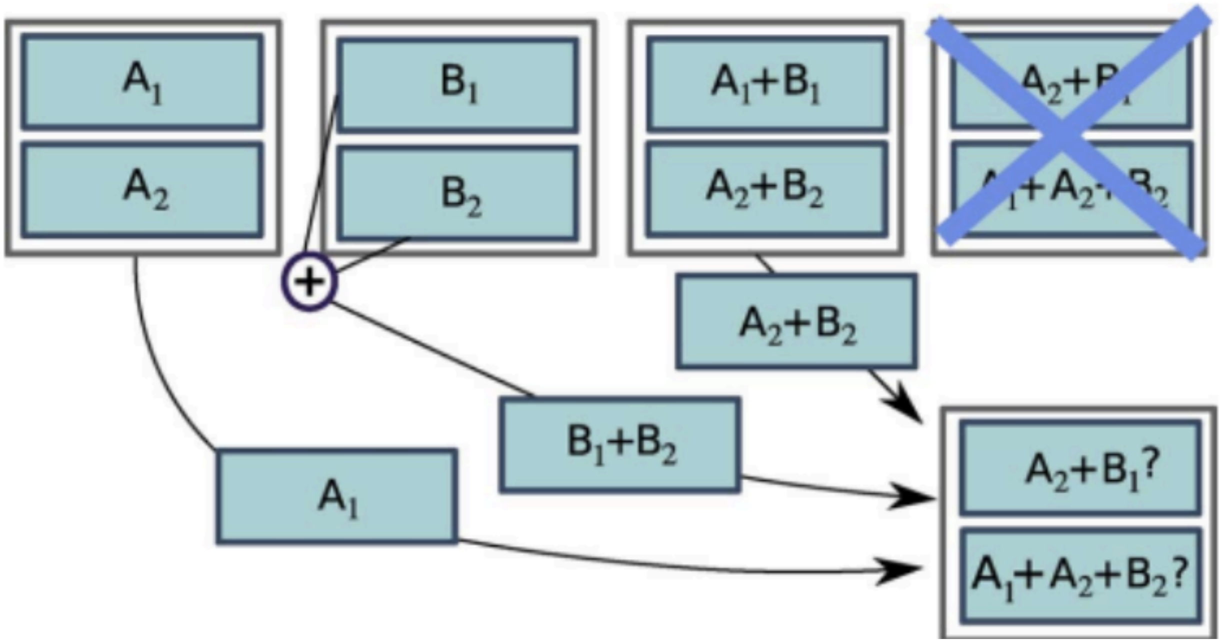


Exact Repair Code Constructions

- For (n,k=2) E-MSR repair can match cutset bound. [WD ISIT'09]
- (n=5,k=3) E-MSR systematic code exists [Cullina,Dimakis, Ho, Allerton'09]
- For **k/n <=1/2** E-MSR repair can match cutset bound [Rashmi, Shah, Kumar, Ramchandran (2010)]
- [Cadambe, Jafar, Maleki] proposed codes to achieve the E-MSR point **for all (k,n,d)**.
- E-MBR for all n,k, for d=n-1 matches cut-set bound [Suh, Ramchandran (2010)]

Locally Repairable Codes

- Codes designed to minimize:
  - Repair Bandwidth
  - **Number of nodes contacted** [Gopalan 2012, Papailiopoulos 2014]



Locally Repairable Codes

- (n, r, d, M, α) LRC
  - Repair a failed node from r other nodes
  - Trade-off between the distance d and locality r

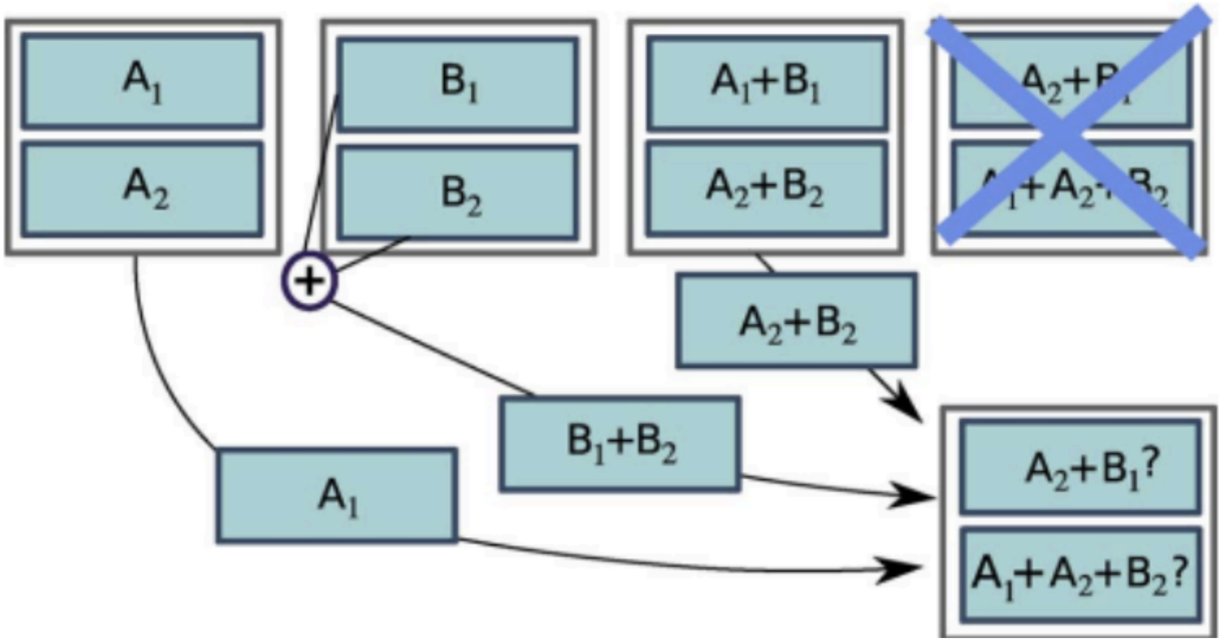
[Papailiopoulos et al 2014]:

Theorem 1. *An (n, r, d, M, α)-LRC has minimum distance d that is bounded as*

$$d \leq n - \left\lceil \frac{M}{\alpha} \right\rceil - \left\lceil \frac{M}{r\alpha} \right\rceil + 2.$$

Data I/O considerations

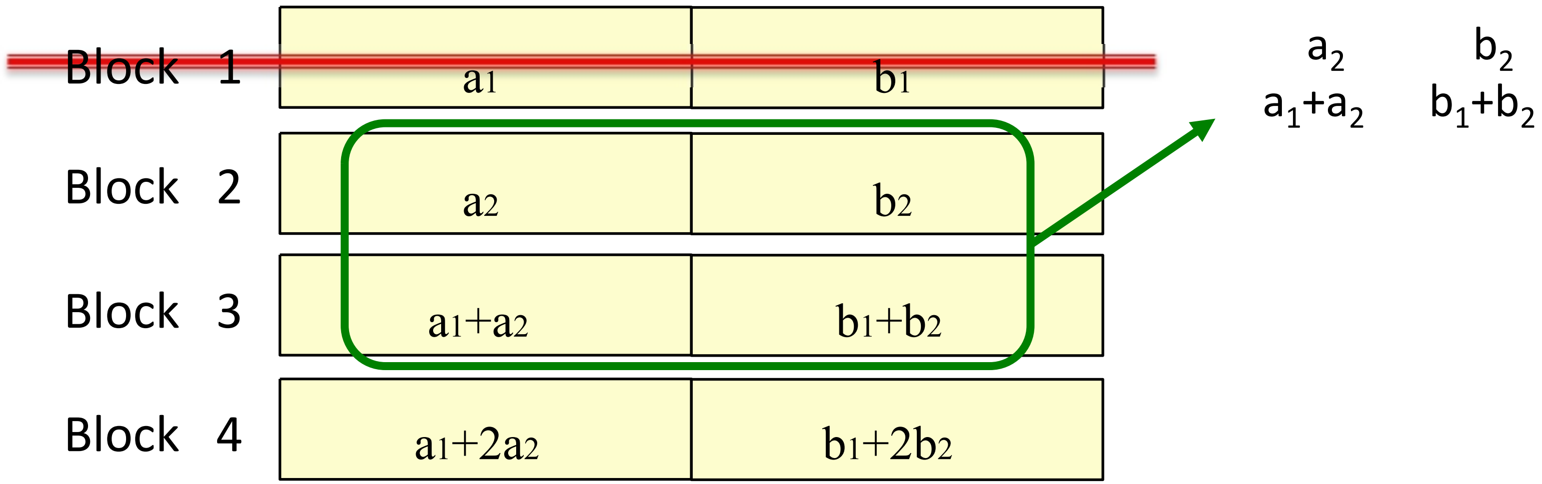
Piggybacking codes [Rashmi et al 2012, 13, 15]



- Data I/O from disk = 4 blocks
- Repair Bandwidth = 3 blocks

Piggybacking Codes

(4,2) Reed-Solomon Code Example



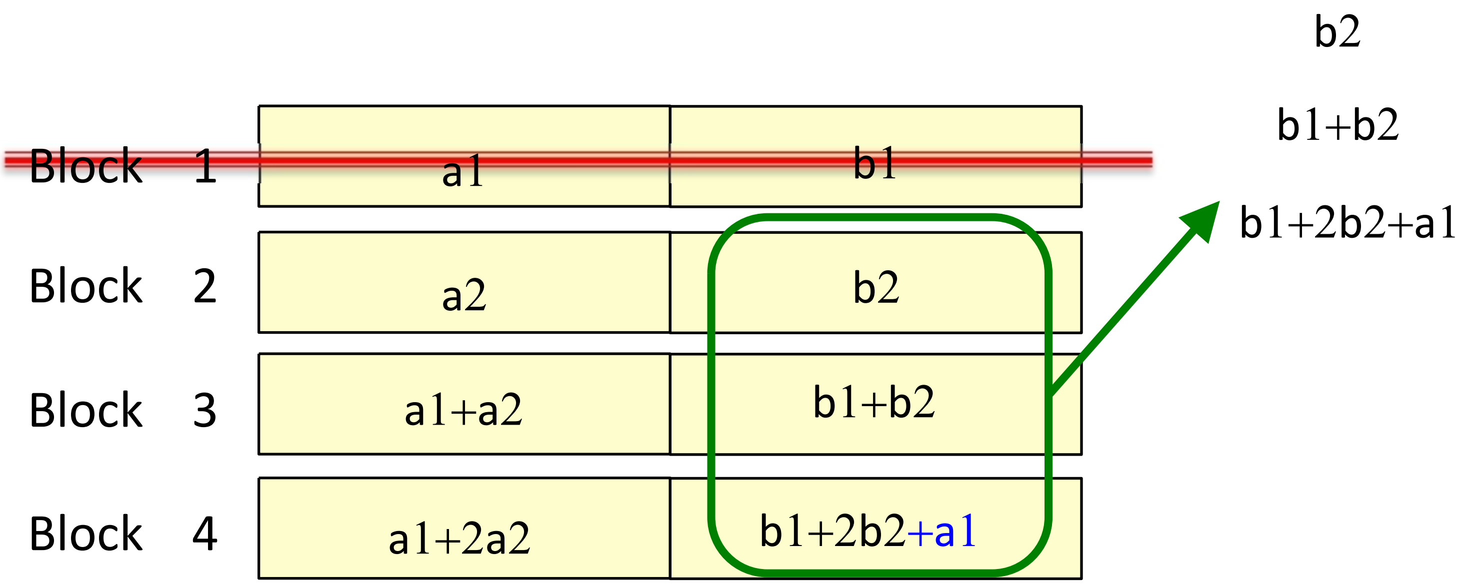
Piggybacking Codes

(4,2) Reed-Solomon Code Example

~~Block 1~~	~~a1~~	~~b1~~
Block 2	a2	b2
Block 3	a1+a2	b1+b2
Block 4	a1+2a2	b1+2b2+a1

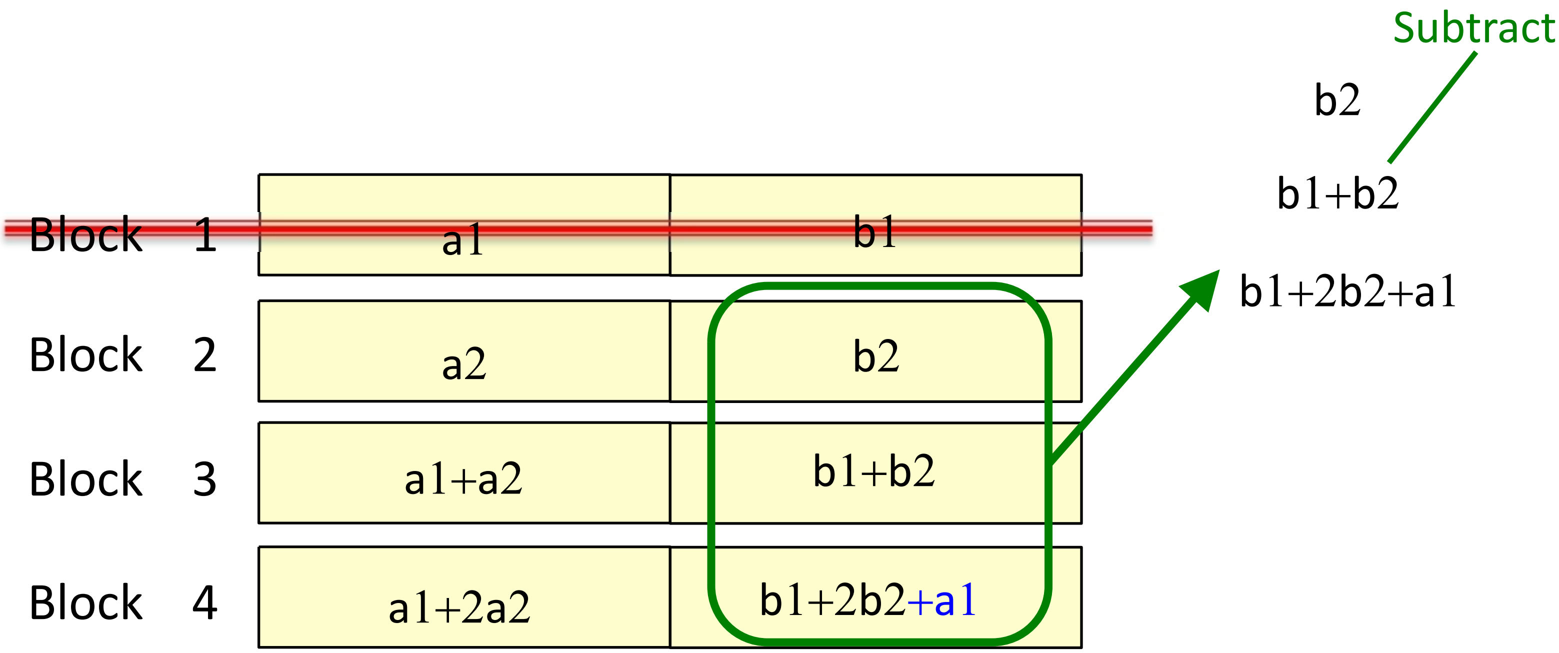
Piggybacking Codes

(4,2) Reed-Solomon Code Example



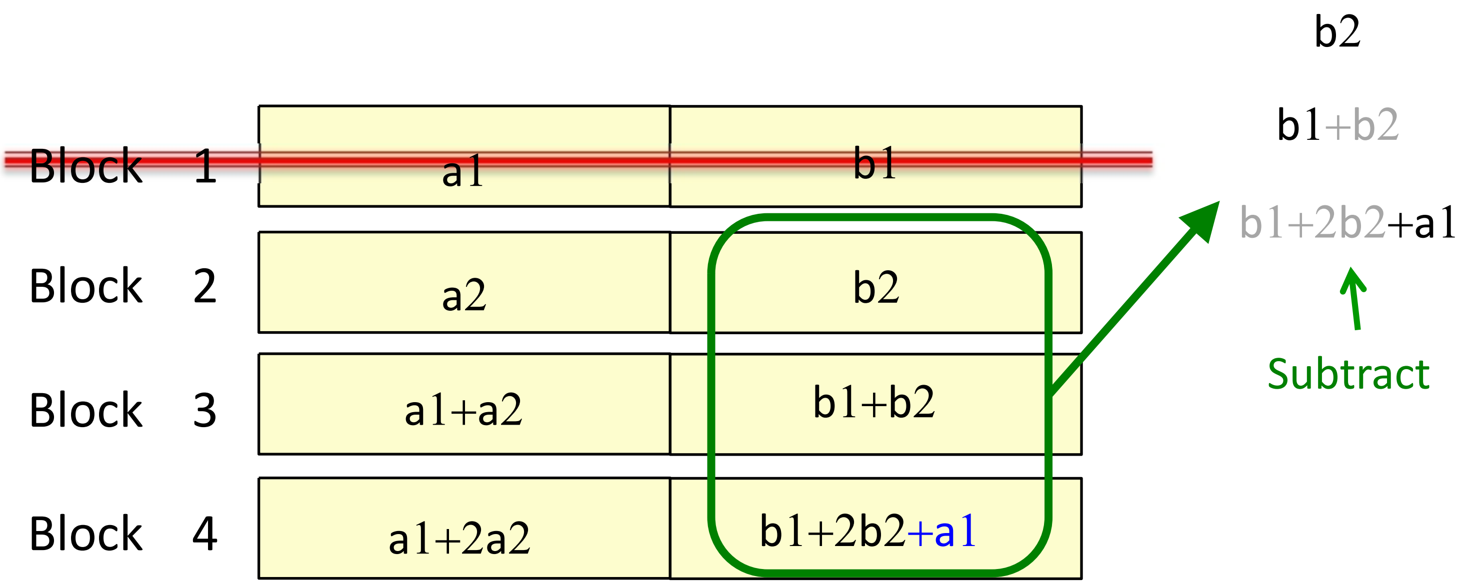
Piggybacking Codes

(4,2) Reed-Solomon Code Example



Piggybacking Codes

(4,2) Reed-Solomon Code Example



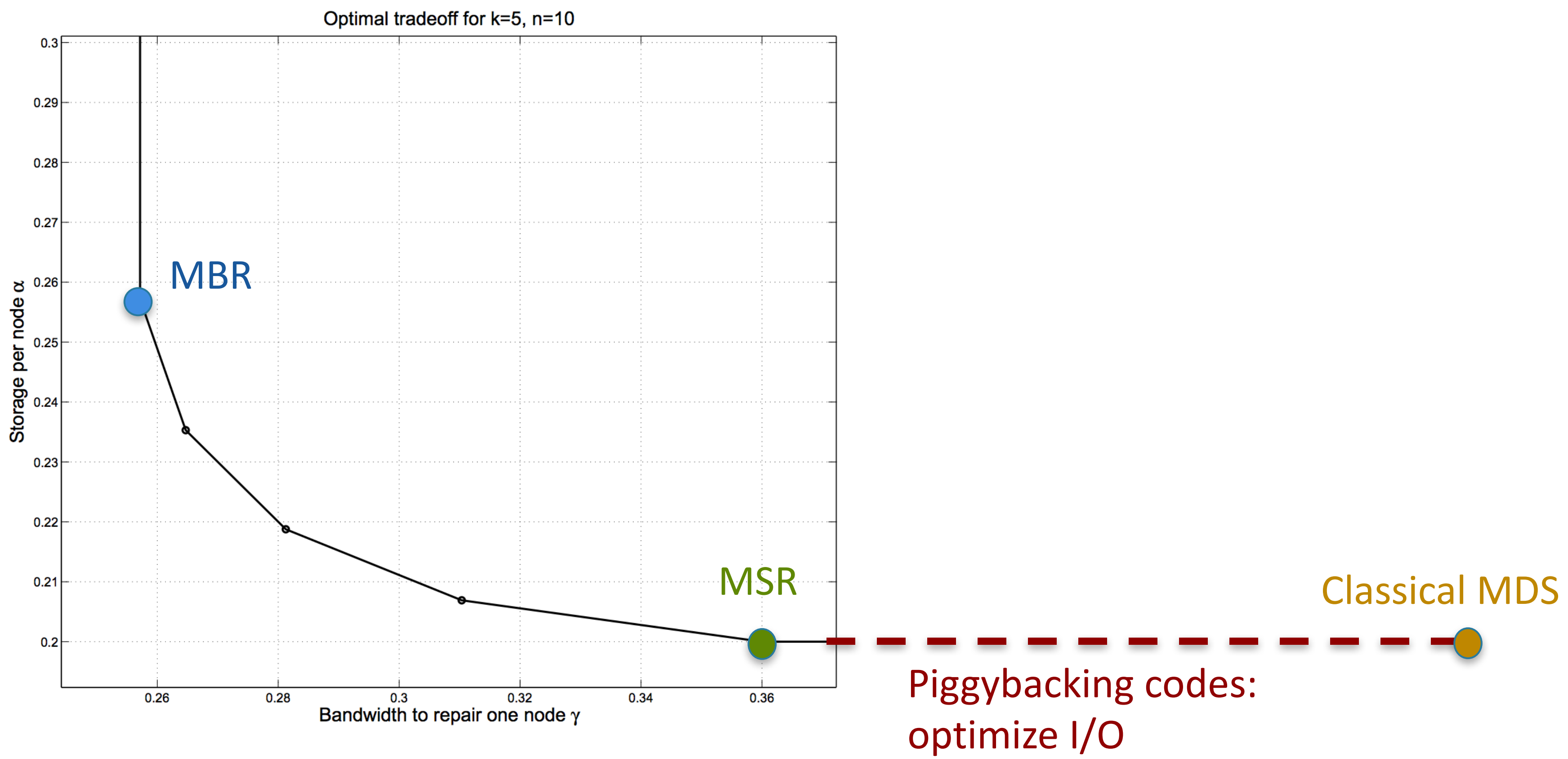
Piggybacking Codes

General Case

Node 1	$f_1(\mathbf{a})$	$f_1(\mathbf{b})$	$\cdots$	$f_1(\mathbf{z})$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Node n	$f_n(\mathbf{a})$	$f_n(\mathbf{b})$	$\cdots$	$f_n(\mathbf{z})$

Node 1	$f_1(\mathbf{a})$	$f_1(\mathbf{b}) + g_{2,1}(\mathbf{a})$	$f_1(\mathbf{c}) + g_{3,1}(\mathbf{a},\mathbf{b})$	$\cdots$	$f_1(\mathbf{z}) + g_{\alpha,1}(\mathbf{a},\ldots,\mathbf{y})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Node n	$f_n(\mathbf{a})$	$f_n(\mathbf{b}) + g_{2,n}(\mathbf{a})$	$f_1(\mathbf{c}) + g_{3,n}(\mathbf{a},\mathbf{b})$	$\cdots$	$f_n(\mathbf{z}) + g_{\alpha,n}(\mathbf{a},\ldots,\mathbf{y})$

Piggybacking Codes



Concept Check: Piggybacking Codes

How many symbols need to be read to repair node 1?

	An MDS Code	
Node 1	a_1	b_1
Node 2	a_2	b_2
Node 3	a_3	b_3
Node 4	a_4	b_4
Node 5	$\sum_{i=1}^{4} a_i$	$\sum_{i=1}^{4} b_i$
Node 6	$\sum_{i=1}^{4} ia_i$	$\sum_{i=1}^{4} ib_i$

(a)

	Intermediate Step	
Node 1	a_1	b_1
Node 2	a_2	b_2
Node 3	a_3	b_3
Node 4	a_4	b_4
Node 5	$\sum_{i=1}^{4} a_i$	$\sum_{i=1}^{4} b_i$
Node 6	$\sum_{i=1}^{4} ia_i$	$\sum_{i=1}^{4} ib_i + \sum_{i=1}^{2} ia_i$

(b)

	Piggybacked Code	
Node 1	a_1	b_1
Node 2	a_2	b_2
Node 3	a_3	b_3
Node 4	a_4	b_4
Node 5	$\sum_{i=1}^{4} a_i$	$\sum_{i=1}^{4} b_i$
Node 6	$\sum_{i=3}^{4} ia_i - \sum_{i=1}^{4} ib_i$	$\sum_{i=1}^{4} ib_i + \sum_{i=1}^{2} ia_i$

(c)

Needs 8 symbols to repair

Needs 6 symbols to repair

Outline

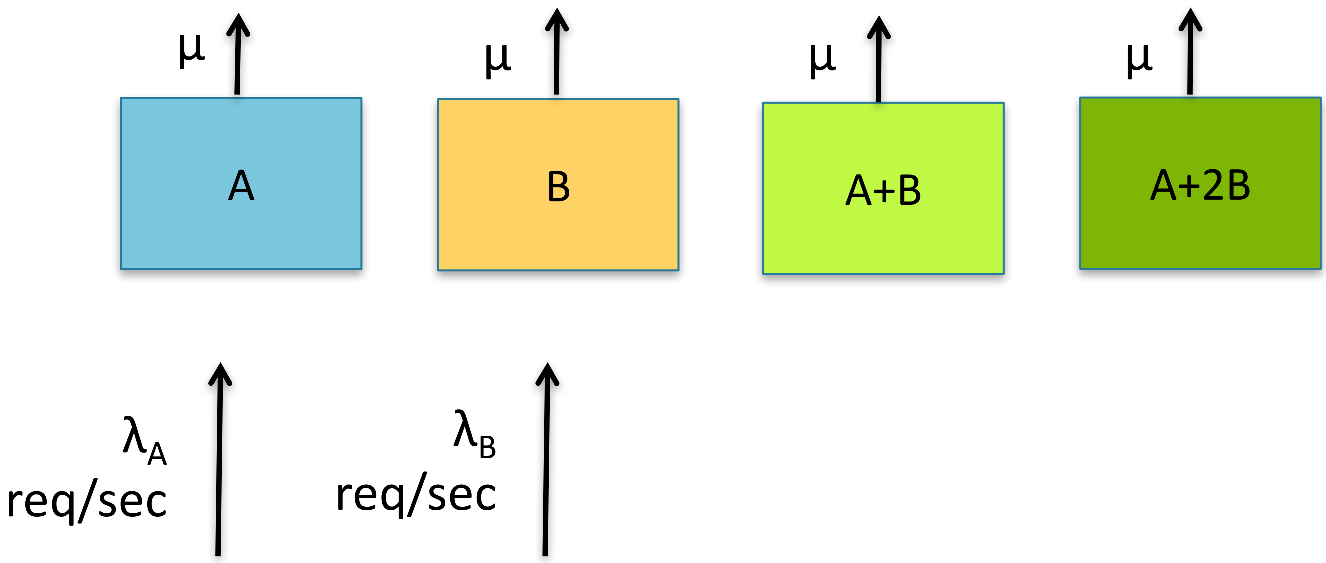
Coded Distributed Storage

Repair-efficiency

Service Capacity

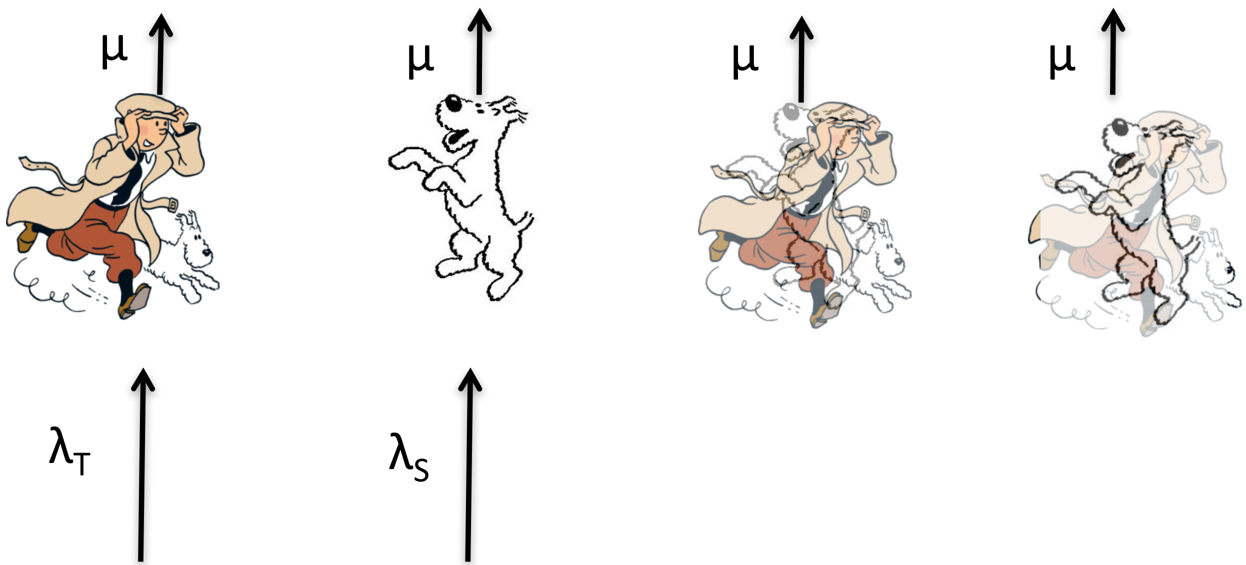
Problem Formulation

- Users may want to access only one of the two chunks
- Applications: Netflix or any content hosting system
- How many requests can we simultaneously support?



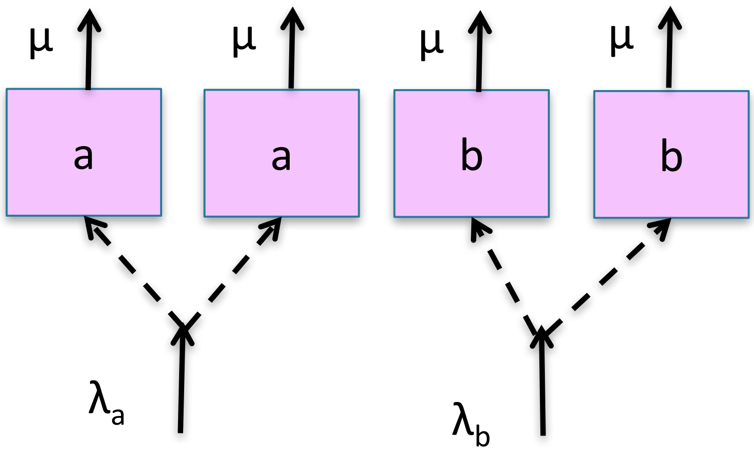
Problem Formulation

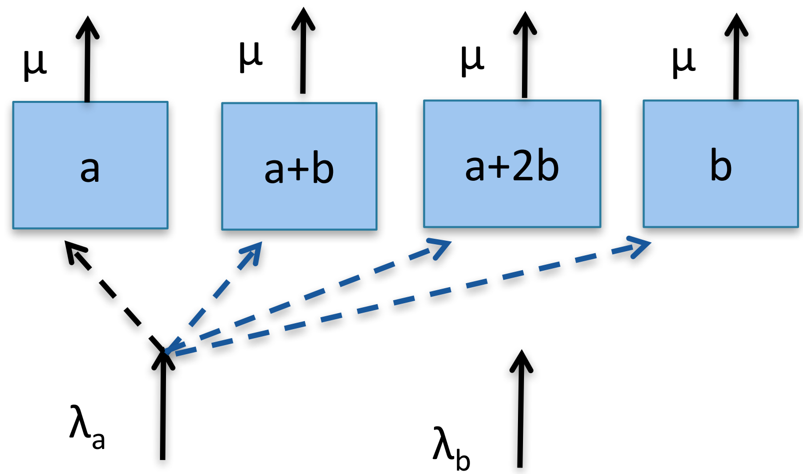
- Users may want to access only one of the two chunks
- Applications: Netflix or any content hosting system
- How many requests can we simultaneously support?

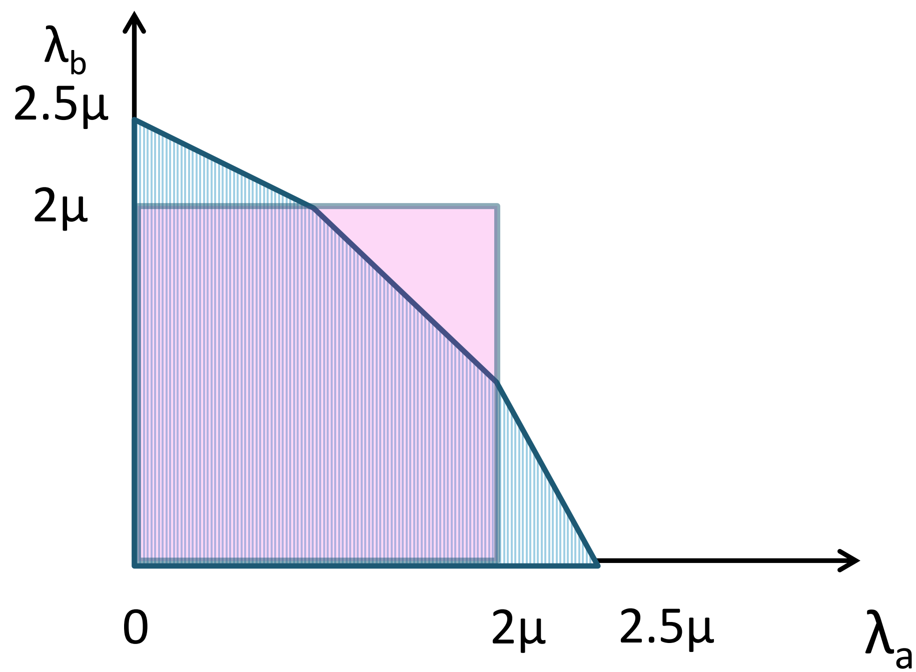


- What is the set of arrival rates (λ_T, λ_s) that we can support?

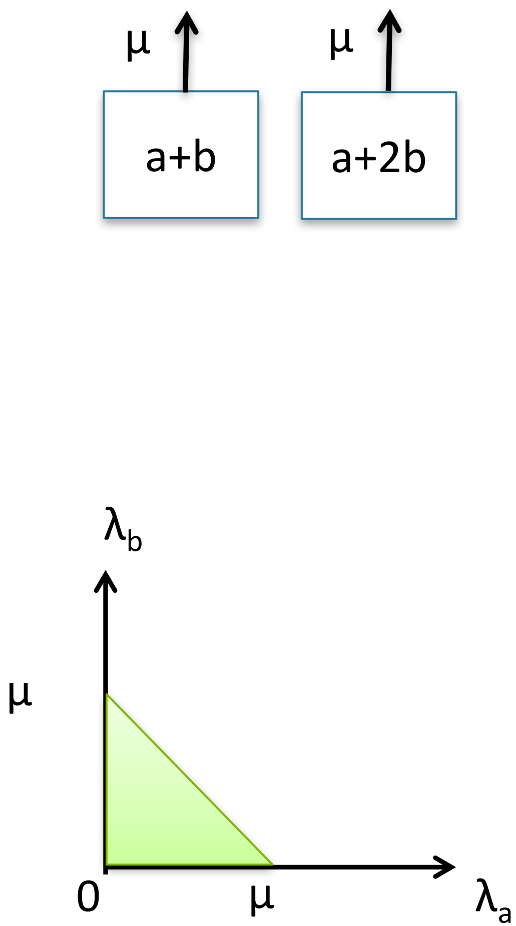
Replication Vs. Coding [Anderson et al 2017]

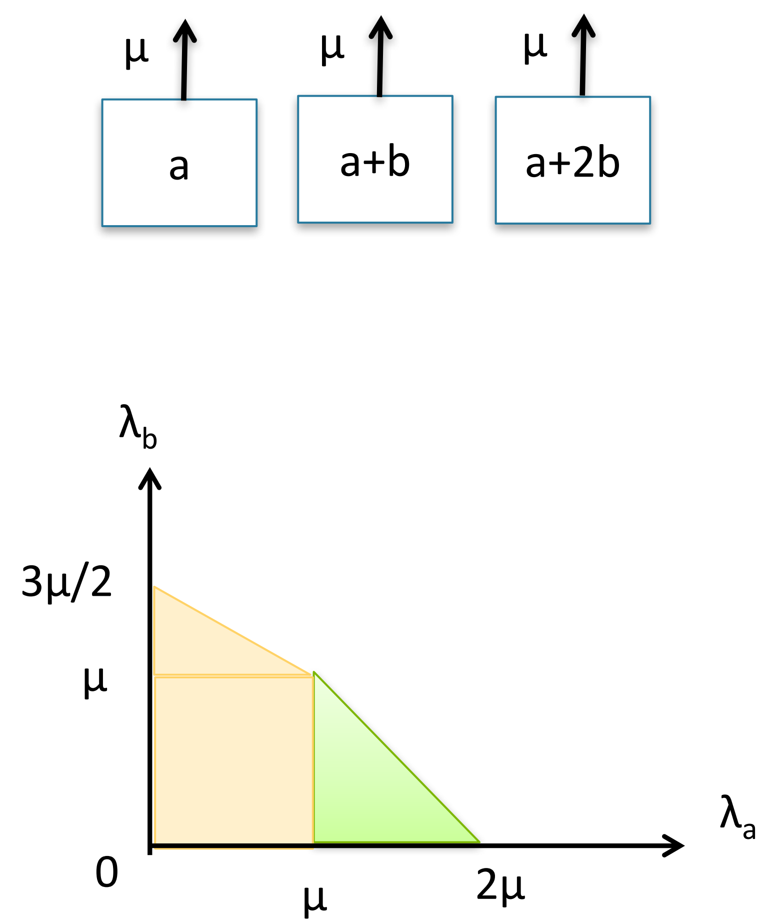






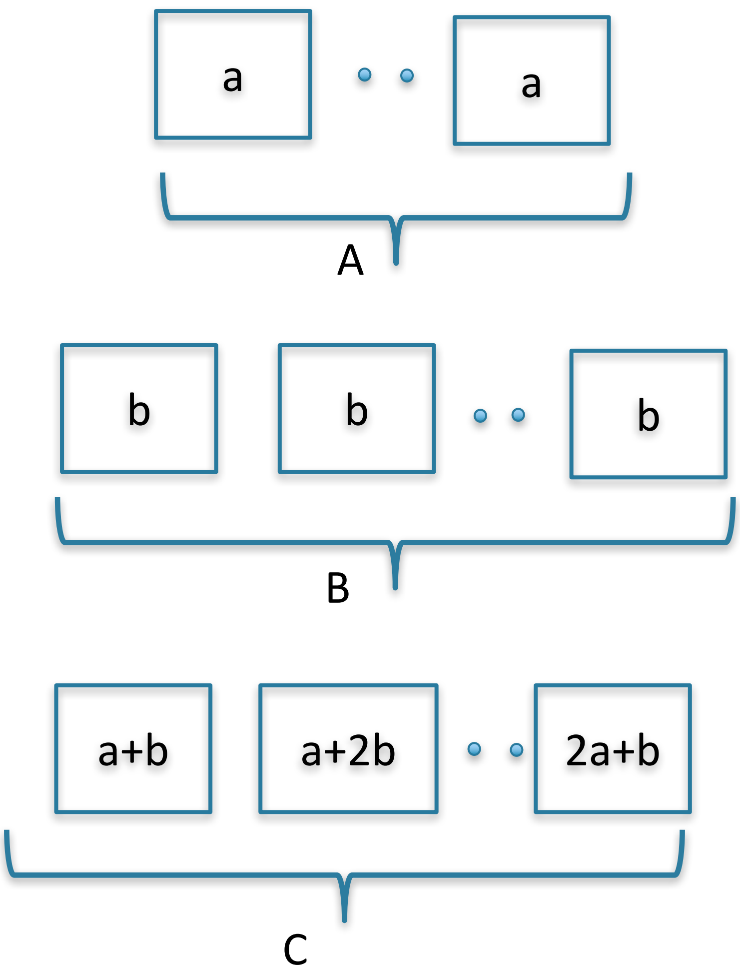
Adding Uncoded Nodes [Anderson et al 2017]





Service Capacity of Coded Storage

[Anderson et al 2017]



Service Capacity Region

Region Widths:

- $(A - C)\mu$ if $A > C$, 0 if $A \leq C$
- $A\mu$ if $A < C$, C if $A \geq C$
- $\frac{C}{2}\mu$
- $\frac{B}{2}\mu$ if $B < C$, $\frac{C}{2}\mu$ if $B \geq C$
- 0

Region Heights:

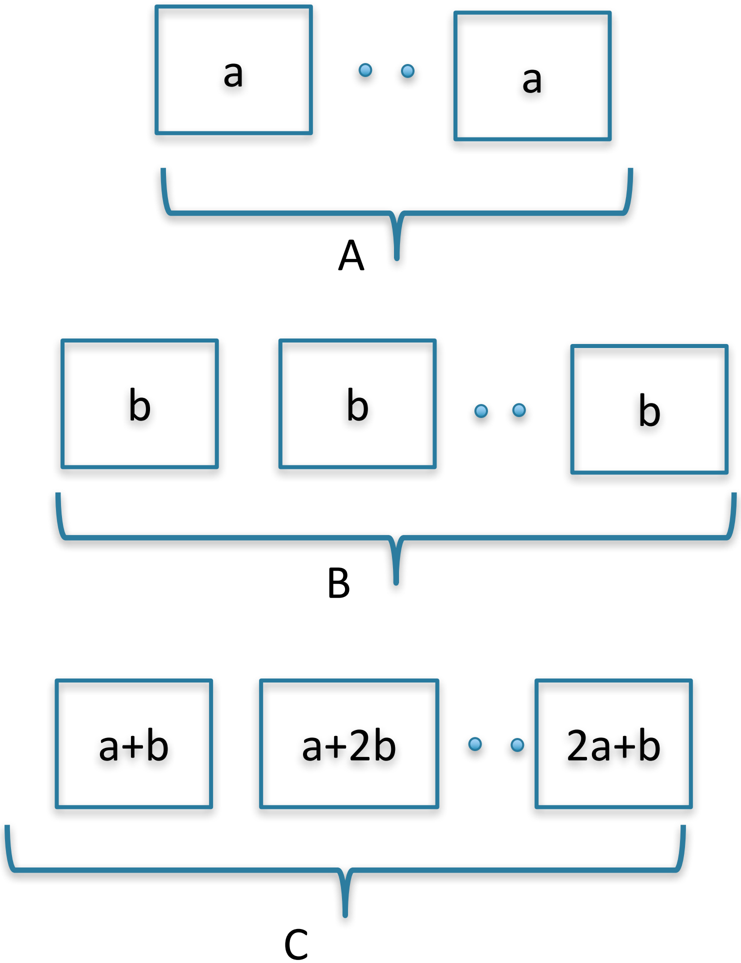
- 0
- $\frac{1}{2}A\mu$ if $A < C$, $\frac{C}{2}\mu$ if $A \geq C$
- $\frac{C}{2}\mu$
- $B\mu$ if $B < C$, C if $B \geq C$
- $(B - C)\mu$ if $B > C$, 0 if $B \leq C$

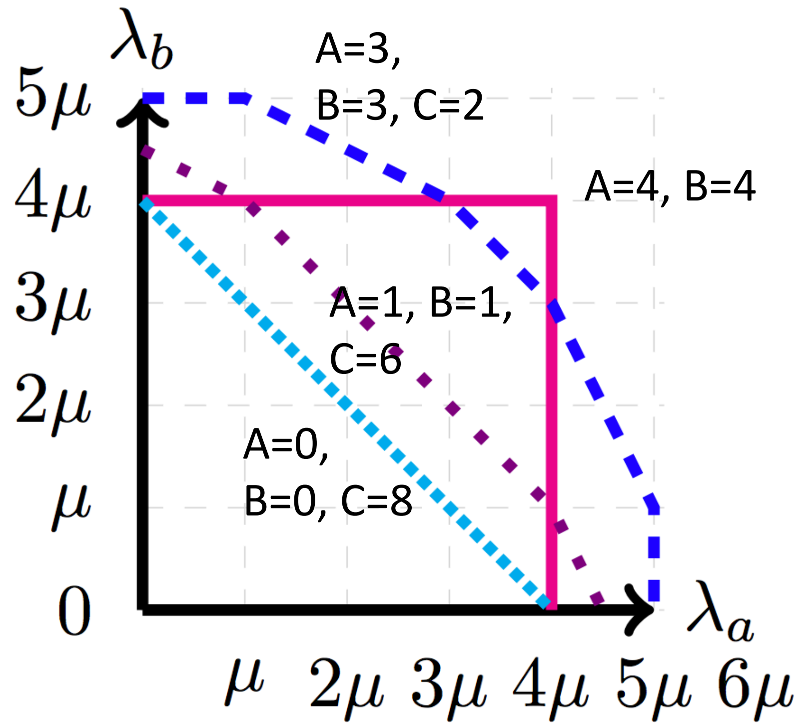
Slopes:

- 0
- $\frac{1}{2}$
- -1
- -2
- vertical

Service Capacity of Coded Storage

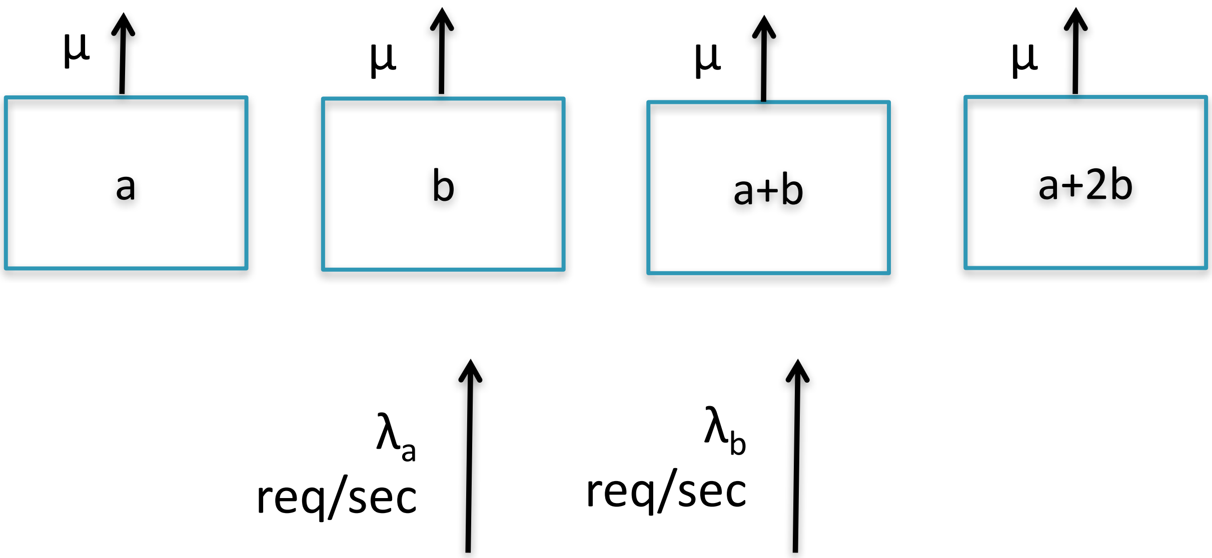
[Anderson et al 2017]





Maximizing Service Capacity: k files, n nodes

- Q1: Given a code, how to optimally split the requests?
- Q2: What is the best underlying erasure code?



Other considerations

Latency

Security

Update-efficiency

Next Lecture: Coded Computing

Approx. Computing

Matrix-vector & matrix-matrix mult.

Distributed Machine Learning