Flexible Coding for Distributed Systems

Zhiying Wang

Joint work with: Weiqi Li, Zhen Chen, Syed A. Jafar, Hamid Jafarkhani
June, 2022
IEEE ComSoc Orange County Chapter

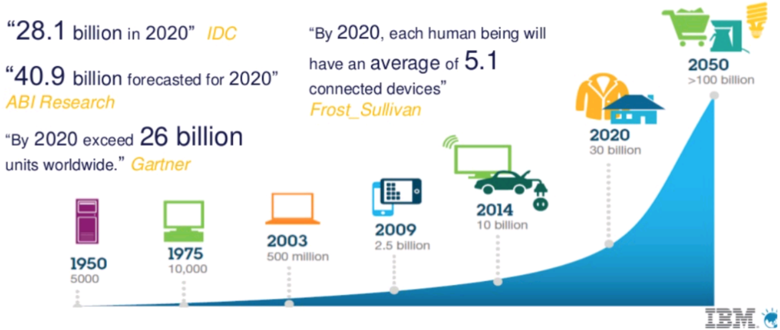
UCI University of California, Irvine

Table of Contents

Table of Contents

Background

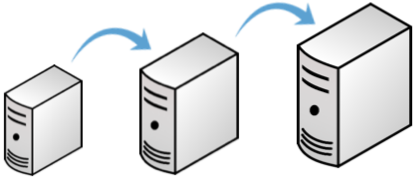
- The amount of data and computation growth exponentially.
- Scaling services: How to address growth?



Background

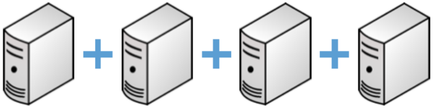
Vertical "Scale up"

- Add more resources to one device.
- Easier, but limited scale.
- Single point of failure.



Horizontal "Scale out"

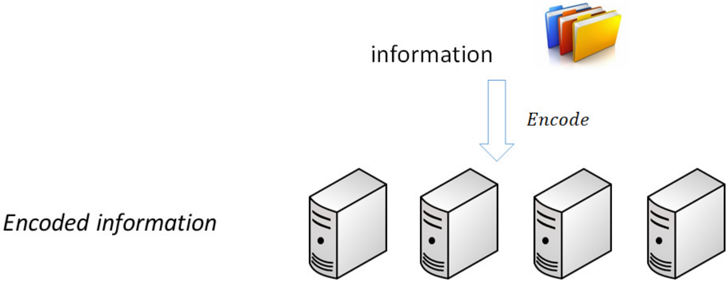
- Run the service over multiple devices.
- Harder, but massive scale.
- Failure tolerance.



R, Appuswamy, C. Gkantsidis, D. Narayanan, O. Hodson, and A. Rowstron, Scale-up vs scale-out for Hadoop: time to rethink?, ACM Symp. Cloud Comput, 2013.

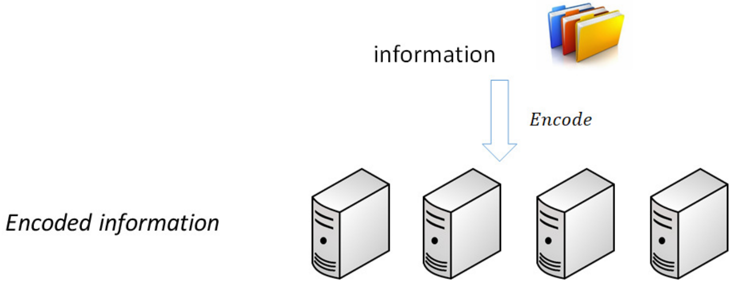
Background

- Distributed systems are widely used for storage and computation.



Background

- Distributed systems are widely used for storage and computation.



- Number of nodes: n.
- Dimension: k.
- Recovery threshold: R.

Background

- Failures are frequent in distributed storage
- This talk: information storage and computing with unknown failures

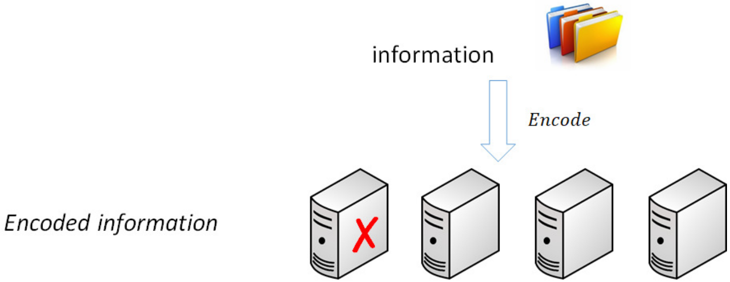
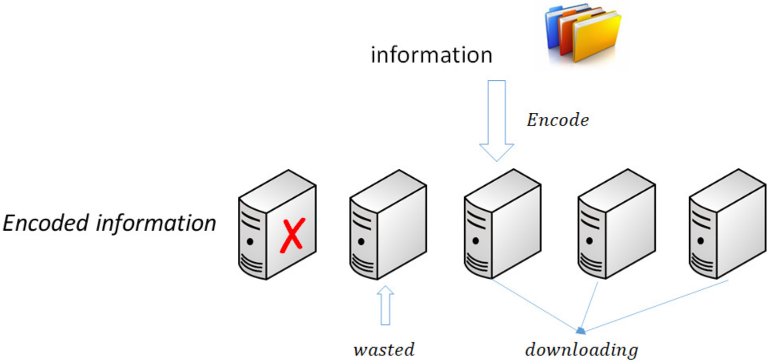


Table of Contents

Motivation

- Fixed code can only make use of R nodes.
- The rest nodes are wasted.
- Each node downloading all symbols $\rightarrow$ large latency.
- Question: storage codes with flexible recovery threshold R?



W. Li, Z. Wang, T. Lu and H. Jafarkhani, Storage Codes with Flexible Number of Nodes, ArXiv:2106.11336, 2021.

Fixed MDS (Maximum Distance Separable) Code

- MDS = minimum redundancy.
- Applied in Google's Colossus, Facebook's f4, Yahoo Object Store, Baidu's Atlas...

Fixed MDS (Maximum Distance Separable) Code

a_1	a_2	$a_1 + a_2$	$a_1 + 2a_2$
b_1	b_2	$b_1 + b_2$	$b_1 + 2b_2$
c_1	c_2	$c_1 + c_2$	$c_1 + 2c_2$

- Example of an $(n, k, \ell) = (4, 2, 3)$ fixed code.
- Each node is a column with $\ell = 3$ symbols.
- $(4, 2)$ MDS code is adopted in each row.

Fixed MDS (Maximum Distance Separable) Code

a_1	a_2	$a_1 + a_2$	$a_1 + 2a_2$
b_1	b_2	$b_1 + b_2$	$b_1 + 2b_2$
c_1	c_2	$c_1 + c_2$	$c_1 + 2c_2$

- Example of an $(n, k, \ell) = (4, 2, 3)$ fixed code.
- Each node is a column with $\ell = 3$ symbols.
- $(4, 2)$ MDS code is adopted in each row.
- 2 failures: 2 nodes send all their symbols.

Fixed MDS (Maximum Distance Separable) Code

a_1	a_2	$a_1 + a_2$	$a_1 + 2a_2$
b_1	b_2	$b_1 + b_2$	$b_1 + 2b_2$
c_1	c_2	$c_1 + c_2$	$c_1 + 2c_2$

- Example of an $(n, k, \ell) = (4, 2, 3)$ fixed code.
- Each node is a column with $\ell = 3$ symbols.
- $(4, 2)$ MDS code is adopted in each row.
- 2 failures: 2 nodes send all their symbols.
- 1 failure: 2 nodes send all their symbols.
- Question: is it possible to use all 3 nodes but each node sends fewer symbols?

Naive Solution

- Example of an $(n, k, \ell) = (4, 2, 3)$ naive flexible code.

$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	W_1
$C_{2,1}$	$C_{2,2}$	$C_{2,3}$	W_2
W_1'	W_2'	W_3'	W_4'

Naive solution

- $(12, 6)$ MDS code is adopted.

Naive Solution

- Example of an $(n, k, \ell) = (4, 2, 3)$ naive flexible code.

$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	W_1
$C_{2,1}$	$C_{2,2}$	$C_{2,3}$	W_2
W'_1	W'_2	W'_3	W'_4

Naive solution

- $(12, 6)$ MDS code is adopted.
- 2 failures: 2 nodes send all their symbols.

Naive Solution

- Example of an $(n, k, \ell) = (4, 2, 3)$ naive flexible code.

$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	W_1
$C_{2,1}$	$C_{2,2}$	$C_{2,3}$	W_2
W_1'	W_2'	W_3'	W_4'

Naive solution

- $(12, 6)$ MDS code is adopted.
- 2 failures: 2 nodes send all their symbols.
- 1 failure: 3 nodes each sending 2 symbols.

Naive Solution

- Example of an $(n, k, \ell) = (4, 2, 3)$ naive flexible code.

$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	W_1
$C_{2,1}$	$C_{2,2}$	$C_{2,3}$	W_2
W_1'	W_2'	W_3'	W_4'

Naive solution

- $(12, 6)$ MDS code is adopted.
- 2 failures: 2 nodes send all their symbols.
- 1 failure: 3 nodes each sending 2 symbols.
- Require a field size of at least $|\mathbb{F}| = n\ell = 12$.

Related work

- [Jafarkhani-Hajiaghayi, 2014], first proposed flexible ideas.
- [Huang-Langberg-Kliewer-Bruck, 2016], flexible secret sharing.
- [Bitar-Rouayheb, 2016], flexible private information retrieval.
- [Tamo-Ye-Barg, 2019], flexible MDS codes, focus on bandwidth instead of access.
- [Ramamoorthy-Tang-Vontobel, 2019], universal decodable matrices for flexible matrix-vector multiplication.

Proposed Solution: flexible MDS Codes

- Example of an $(n, k, \ell) = (4, 2, 3)$ flexible MDS code.

$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	W_1
$C_{2,1}$	$C_{2,2}$	$C_{2,3}$	W_2
W_1'	W_2'	W_3'	W_4'

Scenario *1:*

2 symbols are accessed in *3* nodes.

$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	W_1	W_1'
$C_{2,1}$	$C_{2,2}$	$C_{2,3}$	W_2	W_2'
W_1'	W_2'	W_3'	W_4'	

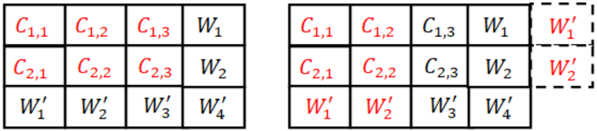
Scenario *2:*

3 symbols are accessed in *2* nodes.

- Row 1: $(5, 3)$ MDS code. W_1, W_1' are parities.
- Row 2: $(5, 3)$ MDS code. W_2, W_2' are parities.
- Row 3: $(4, 2)$ MDS code. W_1', W_2' are information symbols, W_3', W_4' are parities.
- Field size $|\mathbb{F}| = 5$.
- Achieve optimal download of $k\ell = 6$ symbols for 1 or 2 failures.

Proposed Solution: flexible MDS Codes

- Example of an $(n, k, \ell) = (4, 2, 3)$ flexible MDS code.



Scenario *1:* *2* symbols are accessed in *3* nodes.

Scenario *2:* *3* symbols are accessed in *2* nodes.

- General flexible construction: extra parities generated in upper layers and encoded to lower layers.
- Achieve optimal download of $k\ell$ symbols.

Flexible LRC

- LRC (Locally Recoverable Codes): when one node fails, only r helper nodes are accessed.
- High performance in terms of energy and speed.
- Applied in, e.g., Microsoft Azure.
- Optimal LRC codes [Tamo-Barg, 2014] satisfy $R = k + \frac{k}{r} - 1$.
- Question: Flexible recovery threshold R for entire information + locality r for single node recovery?

Flexible LRC

	group 1			$\cdots$	group 4		
Layer 1	$C_{1,1,1}$	$C_{1,1,2}$	$C_{1,1,3}$	$\cdots$	$C_{1,1,10}$	$C_{1,1,11}$	$C_{1,1,12}$
	$C_{1,2,1}$	$C_{1,2,2}$	$C_{1,2,3}$	$\cdots$	$C_{1,2,10}$	$C_{1,2,11}$	$C_{1,2,12}$
Layer 2	$C_{2,1,1}$	$C_{2,1,2}$	$C_{2,1,3}$	$\cdots$	$C_{2,1,10}$	$C_{2,1,11}$	$C_{2,1,12}$

- $(n = 12, k = 4, \ell = 3)$ code. Locality $r = 2$. Recovery threshold $R = 5$.

Flexible LRC

	group 1			$\cdots$	group 4		
Layer 1	$C_{1,1,1}$	$C_{1,1,2}$	$C_{1,1,3}$	$\cdots$	$C_{1,1,10}$	$C_{1,1,11}$	$C_{1,1,12}$
	$C_{1,2,1}$	$C_{1,2,2}$	$C_{1,2,3}$	$\cdots$	$C_{1,2,10}$	$C_{1,2,11}$	$C_{1,2,12}$
Layer 2	$C_{2,1,1}$	$C_{2,1,2}$	$C_{2,1,3}$	$\cdots$	$C_{2,1,10}$	$C_{2,1,11}$	$C_{2,1,12}$

- $(n = 12, k = 4, \ell = 3)$ code. Locality $r = 2$. Recovery threshold $R = 5$.
- Can recover entire $k\ell = 12$ information symbols from:
    - $R_2 = R = 5$ nodes, each accessing $\ell_2 = 3$ symbols
    - $R_1 = 8$ nodes, each accessing $\ell_1 = 2$ symbols
    - Less failures, lower latency

Flexible LRC

	group 1			$\cdots$	group 4		
Layer 1	$C_{1,1,1}$	$C_{1,1,2}$	$C_{1,1,3}$	$\cdots$	$C_{1,1,10}$	$C_{1,1,11}$	$C_{1,1,12}$
	$C_{1,2,1}$	$C_{1,2,2}$	$C_{1,2,3}$	$\cdots$	$C_{1,2,10}$	$C_{1,2,11}$	$C_{1,2,12}$
Layer 2	$C_{2,1,1}$	$C_{2,1,2}$	$C_{2,1,3}$	$\cdots$	$C_{2,1,10}$	$C_{2,1,11}$	$C_{2,1,12}$

- Layer 1: $f_m(x) = \big(u_{m,0} + u_{m,1}g(x) + u_{m,2}g^2(x)\big) + x\big(u_{m,3} + u_{m,4}g(x) + u_{m,5}g^2(x)\big), m = 1, 2.$

Flexible LRC

	group 1			$\cdots$	group 4		
Layer 1	$C_{1,1,1}$	$C_{1,1,2}$	$C_{1,1,3}$	$\cdots$	$C_{1,1,10}$	$C_{1,1,11}$	$C_{1,1,12}$
	$C_{1,2,1}$	$C_{1,2,2}$	$C_{1,2,3}$	$\cdots$	$C_{1,2,10}$	$C_{1,2,11}$	$C_{1,2,12}$
Layer 2	$C_{2,1,1}$	$C_{2,1,2}$	$C_{2,1,3}$	$\cdots$	$C_{2,1,10}$	$C_{2,1,11}$	$C_{2,1,12}$

- Layer 1: $f_m(x) = \big(u_{m,0} + u_{m,1}g(x) + u_{m,2}g^2(x)\big) + x\big(u_{m,3} + u_{m,4}g(x) + u_{m,5}g^2(x)\big), m = 1, 2.$
- Layer 2: $f_3(x) = \big(f_1(\alpha^4) + f_1(\alpha^9)g(x)\big) + x\big(f_2(\alpha^4) + f_2(\alpha^9)g(x)\big).$

Flexible LRC

	group 1			$\cdots$	group 4		
Layer 1	$C_{1,1,1}$	$C_{1,1,2}$	$C_{1,1,3}$	$\cdots$	$C_{1,1,10}$	$C_{1,1,11}$	$C_{1,1,12}$
	$C_{1,2,1}$	$C_{1,2,2}$	$C_{1,2,3}$	$\cdots$	$C_{1,2,10}$	$C_{1,2,11}$	$C_{1,2,12}$
Layer 2	$C_{2,1,1}$	$C_{2,1,2}$	$C_{2,1,3}$	$\cdots$	$C_{2,1,10}$	$C_{2,1,11}$	$C_{2,1,12}$

- Layer 1: $f_m(x) = \big(u_{m,0} + u_{m,1}g(x) + u_{m,2}g^2(x)\big) + x\big(u_{m,3} + u_{m,4}g(x) + u_{m,5}g^2(x)\big), m = 1, 2.$
- Layer 2: $f_3(x) = \big(f_1(\alpha^4) + f_1(\alpha^9)g(x)\big) + x\big(f_2(\alpha^4) + f_2(\alpha^9)g(x)\big).$
- Code over $\mathbb{F} = GF(2^4) = \{0, 1, \alpha, ..., \alpha^{14}\}$. $g(x) = x^3$.
- Evaluated at: $x \in A = \{A_1 = \{1, \alpha^5, \alpha^{10}\}, A_2 = \{\alpha, \alpha^6, \alpha^{11}\}, A_3 = \{\alpha^2, \alpha^7, \alpha^{12}\}, A_4 = \{\alpha^3, \alpha^8, \alpha^{13}\}\}$. Extra group $A_5 = \{\alpha^4, \alpha^9, \alpha^{14}\}$.

Flexible LRC

	group 1			$\cdots$	group 4		
Layer 1	$C_{1,1,1}$	$C_{1,1,2}$	$C_{1,1,3}$	$\cdots$	$C_{1,1,10}$	$C_{1,1,11}$	$C_{1,1,12}$
	$C_{1,2,1}$	$C_{1,2,2}$	$C_{1,2,3}$	$\cdots$	$C_{1,2,10}$	$C_{1,2,11}$	$C_{1,2,12}$
Layer 2	$C_{2,1,1}$	$C_{2,1,2}$	$C_{2,1,3}$	$\cdots$	$C_{2,1,10}$	$C_{2,1,11}$	$C_{2,1,12}$

- Locality: for $x \in A_i$:

$$f_m(x) = \left(u_{m,0} + u_{m,1} + u_{m,2}\right) + x\left(u_{m,3} + u_{m,4} + u_{m,5}\right), m = 1, 2.$$
$$f_3(x) = \left(f_1(\alpha^4) + f_1(\alpha^9)\right) + x\left(f_2(\alpha^4) + f_2(\alpha^9)\right).$$

- All are linear functions of x. $\rightarrow$ Require $r = 2$ evaluations.
- E.g., $A_5 = \{\alpha^4, \alpha^9, \alpha^{14}\}$, $f_1(\alpha^4), f_1(\alpha^9) \rightarrow f_1(\alpha^{14})$

Flexible LRC

	group 1			$\cdots$	group 4		
Layer 1	$C_{1,1,1}$	$C_{1,1,2}$	$C_{1,1,3}$	$\cdots$	$C_{1,1,10}$	$C_{1,1,11}$	$C_{1,1,12}$
	$C_{1,2,1}$	$C_{1,2,2}$	$C_{1,2,3}$	$\cdots$	$C_{1,2,10}$	$C_{1,2,11}$	$C_{1,2,12}$
Layer 2	$C_{2,1,1}$	$C_{2,1,2}$	$C_{2,1,3}$	$\cdots$	$C_{2,1,10}$	$C_{2,1,11}$	$C_{2,1,12}$

- Recovery from $R_1 = 8, \ell_1 = 2$:

$$f_m(x) = \left(u_{m,0} + u_{m,1}g(x) + u_{m,2}g^2(x)\right) + x\left(u_{m,3} + u_{m,4}g(x) + u_{m,5}g^2(x)\right), m = 1, 2.$$

- $f_m(x)$ has degree 7. $(g(x) = x^3)$

Flexible LRC

	group 1			$\cdots$	group 4		
Layer 1	$C_{1,1,1}$	$C_{1,1,2}$	$C_{1,1,3}$	$\cdots$	$C_{1,1,10}$	$C_{1,1,11}$	$C_{1,1,12}$
	$C_{1,2,1}$	$C_{1,2,2}$	$C_{1,2,3}$	$\cdots$	$C_{1,2,10}$	$C_{1,2,11}$	$C_{1,2,12}$
Layer 2	$C_{2,1,1}$	$C_{2,1,2}$	$C_{2,1,3}$	$\cdots$	$C_{2,1,10}$	$C_{2,1,11}$	$C_{2,1,12}$

- Recovery from $R_2 = 5, \ell_2 = 3$:

$$f_m(x) = \left(u_{m,0} + u_{m,1}g(x) + u_{m,2}g^2(x)\right) + x\left(u_{m,3} + u_{m,4}g(x) + u_{m,5}g^2(x)\right), m = 1, 2.$$
$$f_3(x) = \left(f_1(\alpha^4) + f_1(\alpha^9)g(x)\right) + x\left(f_2(\alpha^4) + f_2(\alpha^9)g(x)\right).$$

- $f_3(x)$ has degree 4 $\rightarrow f_1(\alpha^4), f_1(\alpha^9), f_2(\alpha^4), f_2(\alpha^9)$.

Flexible LRC

	group 1			$\cdots$	group 4		
Layer 1	$C_{1,1,1}$	$C_{1,1,2}$	$C_{1,1,3}$	$\cdots$	$C_{1,1,10}$	$C_{1,1,11}$	$C_{1,1,12}$
	$C_{1,2,1}$	$C_{1,2,2}$	$C_{1,2,3}$	$\cdots$	$C_{1,2,10}$	$C_{1,2,11}$	$C_{1,2,12}$
Layer 2	$C_{2,1,1}$	$C_{2,1,2}$	$C_{2,1,3}$	$\cdots$	$C_{2,1,10}$	$C_{2,1,11}$	$C_{2,1,12}$

- Recovery from $R_2 = 5, \ell_2 = 3$:

$$f_m(x) = \left(u_{m,0} + u_{m,1}g(x) + u_{m,2}g^2(x)\right) + x\left(u_{m,3} + u_{m,4}g(x) + u_{m,5}g^2(x)\right), m = 1, 2.$$
$$f_3(x) = \left(f_1(\alpha^4) + f_1(\alpha^9)g(x)\right) + x\left(f_2(\alpha^4) + f_2(\alpha^9)g(x)\right).$$

- $f_3(x)$ has degree 4 $\rightarrow f_1(\alpha^4), f_1(\alpha^9), f_2(\alpha^4), f_2(\alpha^9)$.
- Locality $\rightarrow f_1(\alpha^{14}), f_2(\alpha^{14})$.

Flexible LRC

	group 1			$\cdots$	group 4		
Layer 1	$C_{1,1,1}$	$C_{1,1,2}$	$C_{1,1,3}$	$\cdots$	$C_{1,1,10}$	$C_{1,1,11}$	$C_{1,1,12}$
	$C_{1,2,1}$	$C_{1,2,2}$	$C_{1,2,3}$	$\cdots$	$C_{1,2,10}$	$C_{1,2,11}$	$C_{1,2,12}$
Layer 2	$C_{2,1,1}$	$C_{2,1,2}$	$C_{2,1,3}$	$\cdots$	$C_{2,1,10}$	$C_{2,1,11}$	$C_{2,1,12}$

- Recovery from $R_2 = 5, \ell_2 = 3$:

$$f_m(x) = \left(u_{m,0} + u_{m,1}g(x) + u_{m,2}g^2(x)\right) + x\left(u_{m,3} + u_{m,4}g(x) + u_{m,5}g^2(x)\right), m = 1, 2.$$

$$f_3(x) = \left(f_1(\alpha^4) + f_1(\alpha^9)g(x)\right) + x\left(f_2(\alpha^4) + f_2(\alpha^9)g(x)\right).$$

- $f_3(x)$ has degree 4 $\rightarrow f_1(\alpha^4), f_1(\alpha^9), f_2(\alpha^4), f_2(\alpha^9)$.
- Locality $\rightarrow f_1(\alpha^{14}), f_2(\alpha^{14})$.
- Totally 8 evaluations in Layer 1.

Performance

- Simulation in Amazon Cluster with 8 servers.
- Matrix-vector multiplication is applied.

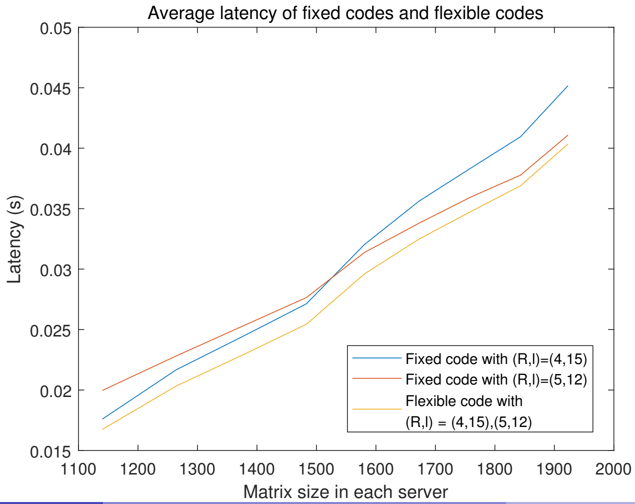
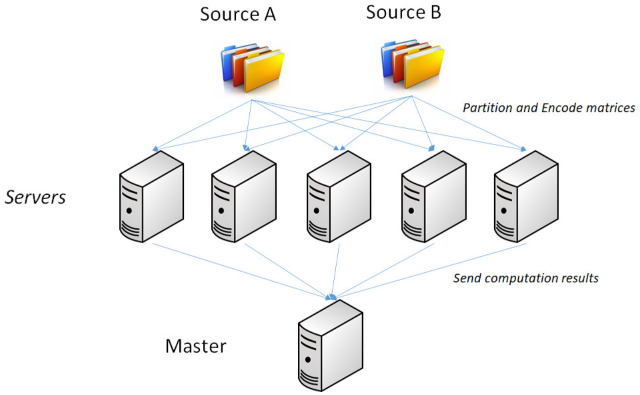


Table of Contents

Motivation

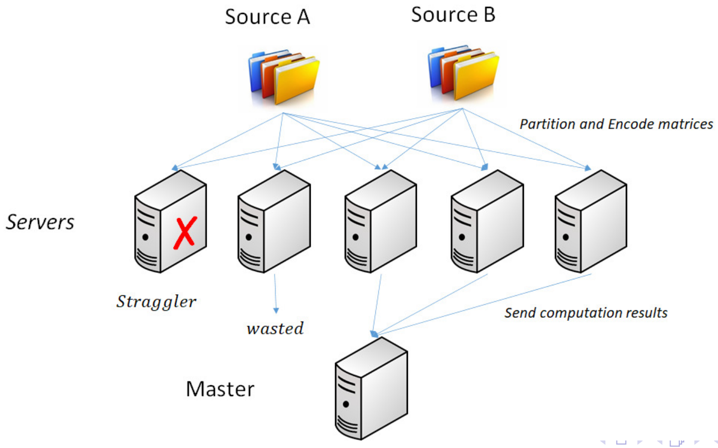
- Matrix multiplication is a central operation of linear algebra.
- Example applications: statistical physics, mathematical finance, machine learning.



- Matrix multiplication: $A \cdot B$.

Motivation

- Fixed code can only make use of R servers.
- The rest available servers are wasted.
- Each available server computes all tasks $\rightarrow$ large latency.



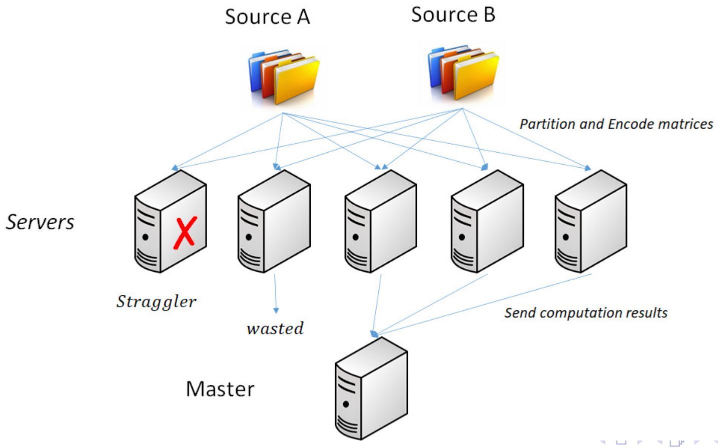
Flexible Matrix Multiplication

- A flexible construction is provided for distributed matrix multiplication and the parameter optimization is analyzed [1]

[1] W Li, Z Chen Z Wang, S.A. Jafar, H Jafarkhani, Flexible Constructions for Distributed Matrix Multiplication, IEEE International Symposium on Information Theory (ISIT) 2021.

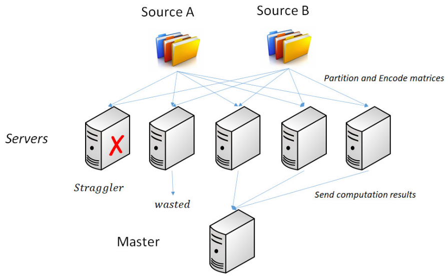
Problem Statement

- Functions of matrix A (and B) are sent to each server.
- Each server performs computation on the functions.
- The master collects computation results and recovers $A \cdot B$.



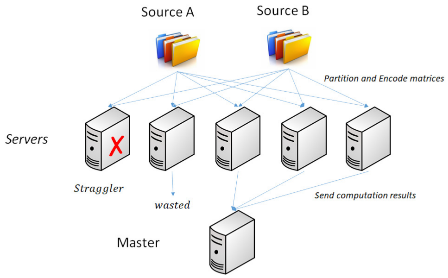
Problem Statement

- Computation load L: the number of multiplications normalized by the total number of multiplications required to multiply two matrices.
- Goal: flexible algorithms with small computation load for unknown stragglers.



Problem Statement

- Computation load L: the number of multiplications normalized by the total number of multiplications required to multiply two matrices.
- Goal: flexible algorithms with small computation load for unknown stragglers.



- Tolerate up to $n - R$ stragglers.
- Stragglers are not known a priori.

Related work

- Coded matrix multiplication with fixed R.

  [Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, ArXiv:1705.10464, 2017], [S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, IEEE Trans IT, 2020], [S. Dutta, Z. Bai, H. Jeong, T. Low, and P. Grover, ArXiv:1811.10751, 2018], [Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, IEEE Trans IT, 2020], [Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. Avestimehr, PMLR, 2019], [Z. Jia and S.A. Jafar, IEEE Trans IT, 2021] ...
- Flexible matrix-vector multiplication.

  [R. Bitar, P. Parag, and S. E. Rouayheb, IEEE Trans Comm, 2020], [R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. E. Rouayheb, and H. Seferoglu, ArXiv:1909.12611, 2019], [A. Ramamoorthy, L. Tang, and P. O. Vontobel, ISIT, 2019], [A. B. Das, L. Tang, and A. Ramamoorthy, ITW, 2018].
- Flexible matrix multiplication with special partition.

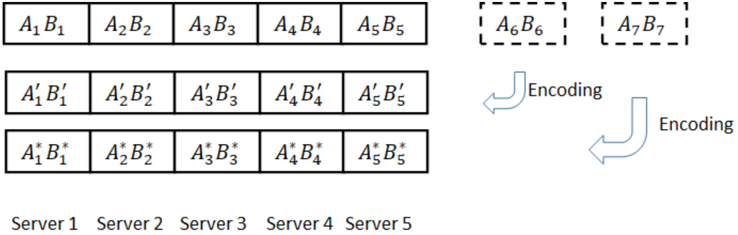
  [R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, ArXiv:2101.05681, 2021], [B. Hasırcıoğlu, J. Gómez-Vilardebó, and D. Gündüz, ArXiv:2001.07227, 2020; Global Comm, 2020], [S. Kiani, N. Ferdinand, and S. C. Draper, ISIT, 2018], [X. Fan, P. Soto, X. Zhong, D. Xi, Y. Wang, and J. Li, IWQoS, 2020], [A. B. Das and A. Ramamoorthy, ArXiv:2012.06065,2020].
- Flexible matrix multiplication with arbitrary partition remains an open problem.

Main Idea

- *Recovery Profile* $\{R_1, \cdots, R_a\}$ instead of recovery threshold R.
- Each server is assigned multiple small subtasks and finishes them sequentially.
- With less stragglers, each server finishes fewer subtasks $\rightarrow$ low latency

Main Idea

- Based on Entangled Polynomial codes[1].
- Extra parities generated in upper layers and encoded to lower layers.

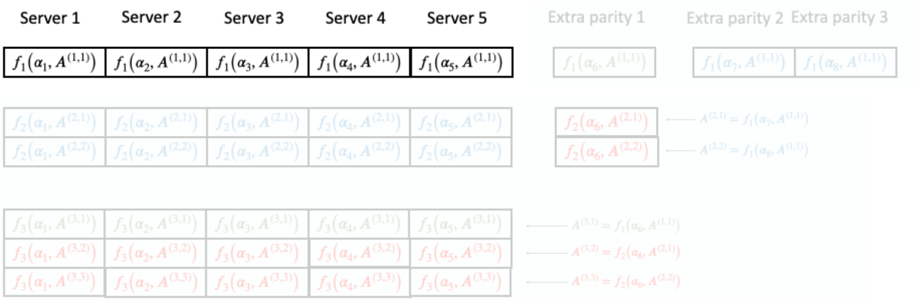


[1] Q. Yu, M.A. Maddah-Ali, A.S. Avestimehr, Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding, IEEE Trans on IT, 2020.

Example

- No stragglers: each server computes 1 task, computation load is $\frac{1}{3}$.

$R_1 = 5, L = \frac{1}{3}$

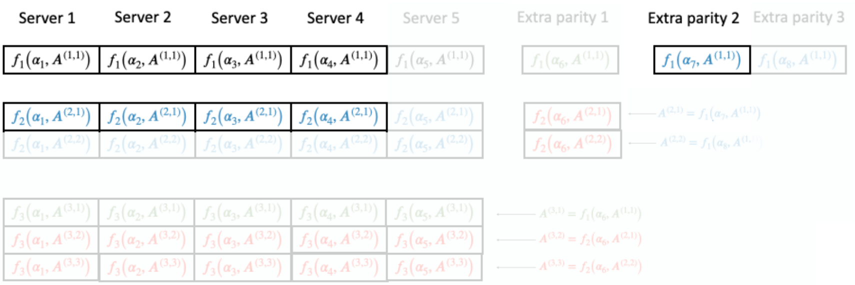


Example

- 1 straggler: each server computes 2 tasks, computation load is $\frac{1}{2}$.

$R_1 = 5, L = \frac{1}{3}$

$R_2 = 3, L = \frac{1}{6}$



Example

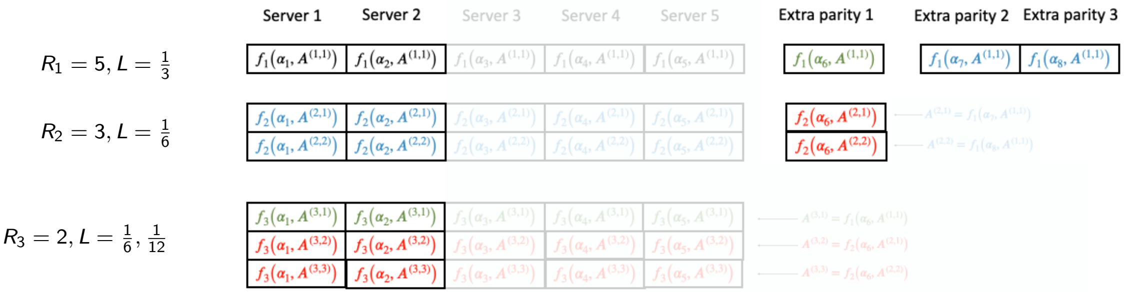
- 2 stragglers: each server computes 3 tasks, computation load is $\frac{2}{3}$.

$R_1 = 5, L = \frac{1}{3}$

$R_2 = 3, L = \frac{1}{6}$

Example

- 3 stragglers: each server computes 6 tasks, computation load is 1.



Performance

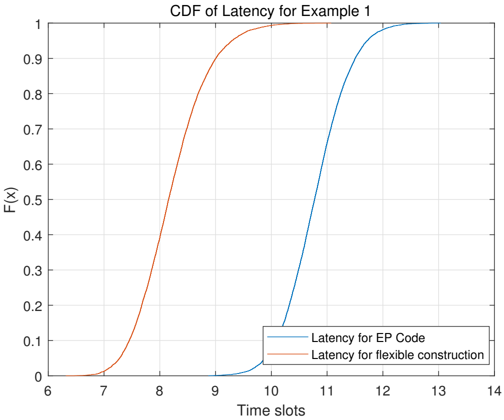
- Let $\hat{R}$ be the number of available servers, $\hat{R} = 2, 3, 4, 5$.
- Let $p(\hat{R})$ be the probability of $\hat{R}$ available servers.
- Expectation over the realizations of $\hat{R}$,

$$E[L_{\text{flex}}] = \sum_{i=2}^{5} p(\hat{R} = i) L_{\text{flex}}(\hat{R} = i).$$

- $p(5) = 0.7, p(2) = p(3) = p(4) = 0.1$, $E[L_{\text{flex}}] = 0.45$, $E[L_{\text{EP}}] = 0.5$.

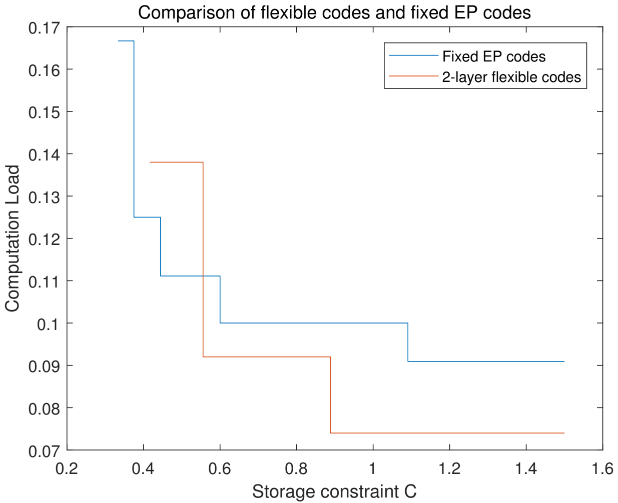
Performance

- Assume 5 servers. Each unit task in each server satisfies an exponential distribution.



Performance

- Assume $n = 16, R_1 = 15, R_2 = R = 11$. 10% straggler probability for each server.



Example

- $n = 5, R = 2,\ \{R_1, R_2, R_3\} = \{5, 3, 2\}$.
- Partition:

$$A = [A_1, A_2, A_3], B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

Example

- $n = 5, R = 2, \{R_1, R_2, R_3\} = \{5, 3, 2\}$.
- Partition:

$$A = [A_1, A_2, A_3], B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

- Encode:

$$A_1 + \alpha_i A_2 + \alpha_i^2 A_3, \quad \alpha_i^2 B_1 + \alpha_i B_2 + B_3.$$

Example

- $n = 5, R = 2, \{R_1, R_2, R_3\} = \{5, 3, 2\}$.
- Partition:

$$A = [A_1, A_2, A_3], B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

- Encode:

$$A_1 + \alpha_i A_2 + \alpha_i^2 A_3, \quad \alpha_i^2 B_1 + \alpha_i B_2 + B_3.$$

- Layer 1 calculates:

$$A_1 B_3 + \alpha_i (A_2 B_3 + A_1 B_2) + \alpha_i^2 (A_1 B_1 + A_2 B_2 + A_3 B_3) + \alpha_i^3 (A_2 B_1 + A_3 B_2) + \alpha_i^4 A_3 B_1.$$

Example

- $n = 5, R = 2, \{R_1, R_2, R_3\} = \{5, 3, 2\}$.
- Partition:

$$A = [A_1, A_2, A_3], B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

- Encode:

$$A_1 + \alpha_i A_2 + \alpha_i^2 A_3, \quad \alpha_i^2 B_1 + \alpha_i B_2 + B_3.$$

- Layer 1 calculates:

$$A_1 B_3 + \alpha_i (A_2 B_3 + A_1 B_2) + \alpha_i^2 (A_1 B_1 + A_2 B_2 + A_3 B_3) + \alpha_i^3 (A_2 B_1 + A_3 B_2) + \alpha_i^4 A_3 B_1.$$

- If no stragglers, computation completes.

Example

- Layer 2: Handle the parities in Layer 1.

$$A_{\alpha_7} = (A_1 + \alpha_7 A_2 + \alpha_7^2 A_3), B_{\alpha_7} = (\alpha_7^2 B_1 + \alpha_7 B_2 + B_3).$$

Example

- Layer 2: Handle the parities in Layer 1.

$$A_{\alpha_7} = (A_1 + \alpha_7 A_2 + \alpha_7^2 A_3), B_{\alpha_7} = (\alpha_7^2 B_1 + \alpha_7 B_2 + B_3).$$

- Partition:

$$A_{\alpha_7} = [A_1', A_2'], B_{\alpha_7} = \begin{bmatrix} B_1' \\ B_2' \end{bmatrix}.$$

- Encode:

$$A_1' + \alpha_i A_2', \quad \alpha_i B_1' + B_2'.$$

Example

- Layer 2: Handle the parities in Layer 1.

$$A_{\alpha_7} = (A_1 + \alpha_7 A_2 + \alpha_7^2 A_3), B_{\alpha_7} = (\alpha_7^2 B_1 + \alpha_7 B_2 + B_3).$$

- Partition:

$$A_{\alpha_7} = [A_1', A_2'], B_{\alpha_7} = \begin{bmatrix} B_1' \\ B_2' \end{bmatrix}.$$

- Encode:

$$A_1' + \alpha_i A_2', \quad \alpha_i B_1' + B_2'.$$

- Layer 2 calculates:

$$A_1' B_2' + \alpha_i (A_1' B_1' + A_2' B_2') + \alpha_i^2 A_2' B_1'.$$

Example

- Layer 2: Handle the parities in Layer 1.

$$A_{\alpha_7} = (A_1 + \alpha_7 A_2 + \alpha_7^2 A_3), B_{\alpha_7} = (\alpha_7^2 B_1 + \alpha_7 B_2 + B_3).$$

- Partition:

$$A_{\alpha_7} = [A_1', A_2'], B_{\alpha_7} = \begin{bmatrix} B_1' \\ B_2' \end{bmatrix}.$$

- Encode:

$$A_1' + \alpha_i A_2', \quad \alpha_i B_1' + B_2'.$$

- Layer 2 calculates:

$$A_1' B_2' + \alpha_i (A_1' B_1' + A_2' B_2') + \alpha_i^2 A_2' B_1'.$$

- Same for $A_{\alpha_8}, B_{\alpha_8}$.
- More matrices are sent to servers, while the matrices are smaller.

Example

- Layer 3: Handle the parities in Layer 1 and 2.

$$A^{(3,1)} = f_1(\alpha_6, A^{(1,1)}), \quad A^{(3,2)} = f_2(\alpha_6, A^{(2,1)}), \quad A^{(3,3)} = f_2(\alpha_6, A^{(2,2)})$$
$$B^{(3,1)} = g_1(\alpha_6, B^{(1,1)}), \quad B^{(3,2)} = g_2(\alpha_6, B^{(2,1)}), \quad B^{(3,3)} = g_2(\alpha_6, B^{(2,2)})$$

- Partition: $A^{(3,1)} = A_1'', B^{(3,1)} = [B_1'', B_2''].$

Example

- Layer 3: Handle the parities in Layer 1 and 2.

$$A^{(3,1)} = f_1(\alpha_6, A^{(1,1)}), \quad A^{(3,2)} = f_2(\alpha_6, A^{(2,1)}), \quad A^{(3,3)} = f_2(\alpha_6, A^{(2,2)})$$
$$B^{(3,1)} = g_1(\alpha_6, B^{(1,1)}), \quad B^{(3,2)} = g_2(\alpha_6, B^{(2,1)}), \quad B^{(3,3)} = g_2(\alpha_6, B^{(2,2)})$$

- Partition: $A^{(3,1)} = A_1'', B^{(3,1)} = [B_1'', B_2'']$.
- Encode:

$$A_1'', \quad g_3(\alpha_i, \alpha_i B_1'' + B_2''.$$

- Layer 3 calculates:

$$A_1'' B_2'' + \alpha_i A_1'' B_1''.$$

- Same for $A^{(3,2)}, B^{(3,2)}, A^{(3,3)}, B^{(3,3)}$.

Optimization

- How to set the parameters
  - number of layers
  - recovery profile
  - partitioning parameters

$$A = \begin{bmatrix} A_{(1,1)} & \cdots & A_{(1,p)} \\ A_{(2,1)} & \cdots & A_{(2,p)} \\ \vdots & \vdots & \vdots \\ A_{(m,1)} & \cdots & A_{(m,p)} \end{bmatrix}, B = \begin{bmatrix} B_{(1,1)} & \cdots & B_{(1,n)} \\ B_{(2,1)} & \cdots & B_{(2,n)} \\ \vdots & \vdots & \vdots \\ B_{(p,1)} & \cdots & B_{(p,n)} \end{bmatrix}.$$

Optimization

- Minimize the expectation

$$E[L_{\text{flex}}] = \sum_{i=R}^{n} p(\hat{R} = i) L_{\text{flex}}(\hat{R} = i),$$

  - Over the number of layers a.
  - Over recovery profile $\{R_1, \cdots, R_a\}$.
  - Over the partitioning parameters $p_j, m_j, n_j, j \in [a]$.

Optimization

- Theorem. When the probability of no straggler is large enough, the maximum number of layers is optimal.
  - $a = n - R + 1$ and recovery profile $\{R_1, \cdots, R_a\} = \{n, n-1, \cdots, R\}$.

Optimization

- Theorem. When the probability of no straggler is large enough, the maximum number of layers is optimal.
  - $a = n - R + 1$ and recovery profile $\{R_1, \cdots, R_a\} = \{n, n-1, \cdots, R\}$.
  - p_1 is an integer around $\frac{1}{2}(R+1) - \frac{1}{2}\sqrt{(R+1)^2 - (16\lambda\kappa^2\mu)/C^2}$.
  - $p_j = 1, m_j n_j = R_j$, $j \geq 2$, matrix-vector multiplication.

Optimization

- Theorem. When the probability of no straggler is large enough, the maximum number of layers is optimal.
    - $a = n - R + 1$ and recovery profile $\{R_1, \cdots, R_a\} = \{n, n-1, \cdots, R\}$.
    - p_1 is an integer around $\frac{1}{2}(R+1) - \frac{1}{2}\sqrt{(R+1)^2 - (16\lambda\kappa^2\mu)/C^2}$.
    - $p_j = 1, m_j n_j = R_j$, $j \geq 2$, matrix-vector multiplication.
- Key steps:
    - Optimize p_j, m_j, n_j given recovery profile.
    - Given R_1 and R, the more layers, the better.
    - Find the sufficient condition to set $R_1 = n$.

Optimization

- Theorem. When the probability of no straggler is large enough, the maximum number of layers is optimal.
  - $a = n - R + 1$ and recovery profile $\{R_1, \cdots, R_a\} = \{n, n-1, \cdots, R\}$.
  - p_1 is an integer around $\frac{1}{2}(R+1) - \frac{1}{2}\sqrt{(R+1)^2 - (16\lambda\kappa^2\mu)/C^2}$.
  - $p_j = 1, m_j n_j = R_j, j \geq 2$, matrix-vector multiplication.

- $n = 50, R = 40$ and assume the number of stragglers follows a truncated binomial distribution with parameter ϵ. Then $a = n - R + 1 = 11$ layers is optimal if $\epsilon < 7.4\%$.

Table of Contents

Conclusion

- Flexible constructions and optimizations for distributed storage and computing.

Conclusion

- Flexible constructions and optimizations for distributed storage and computing.
- Our flexible storage codes can be generalized to optimal flexible codes that tolerates mixed types of failures (useful for flash drives and RAID) and minimizes the traffic during single-node repair (useful for networked storage).

Conclusion

- Flexible constructions and optimizations for distributed storage and computing.
- Our flexible storage codes can be generalized to optimal flexible codes that tolerates mixed types of failures (useful for flash drives and RAID) and minimizes the traffic during single-node repair (useful for networked storage).
- Our flexible matrix multiplication can be generalized to batch processing and secure distributed matrix multiplication.

Conclusion

- Flexible constructions and optimizations for distributed storage and computing.
- Our flexible storage codes can be generalized to optimal flexible codes that tolerates mixed types of failures (useful for flash drives and RAID) and minimizes the traffic during single-node repair (useful for networked storage).
- Our flexible matrix multiplication can be generalized to batch processing and secure distributed matrix multiplication.
- It is worthwhile to explore more applications of flexible constructions, such as federated learning and secure multi-party computation.

Thank you!