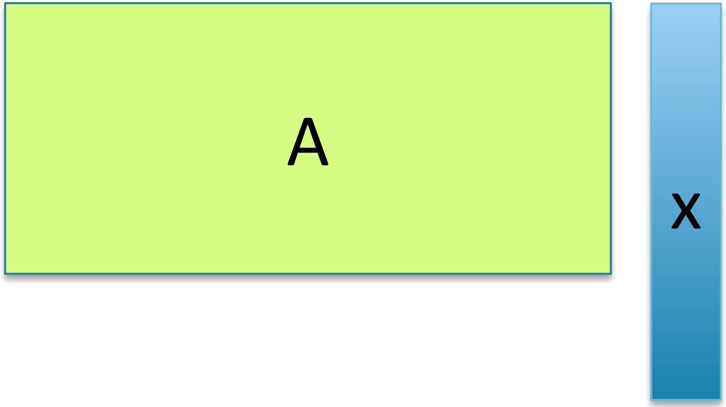
Rateless Codes for Straggler Mitigation in Distributed Computing

Ankur Mallick, Malhar Chaudhari, Gauri Joshi

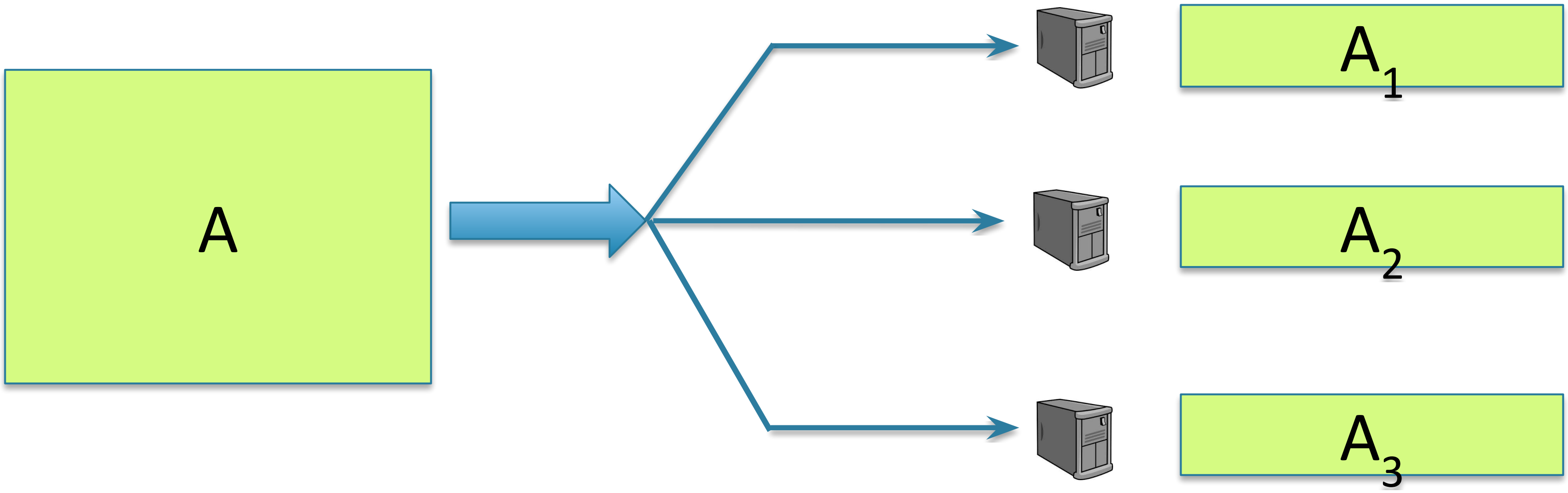
Coded Computing

- What computing jobs can be coded such as any k out of n tasks are sufficient to complete the job?
- Example: Matrix-Vector Multiplication



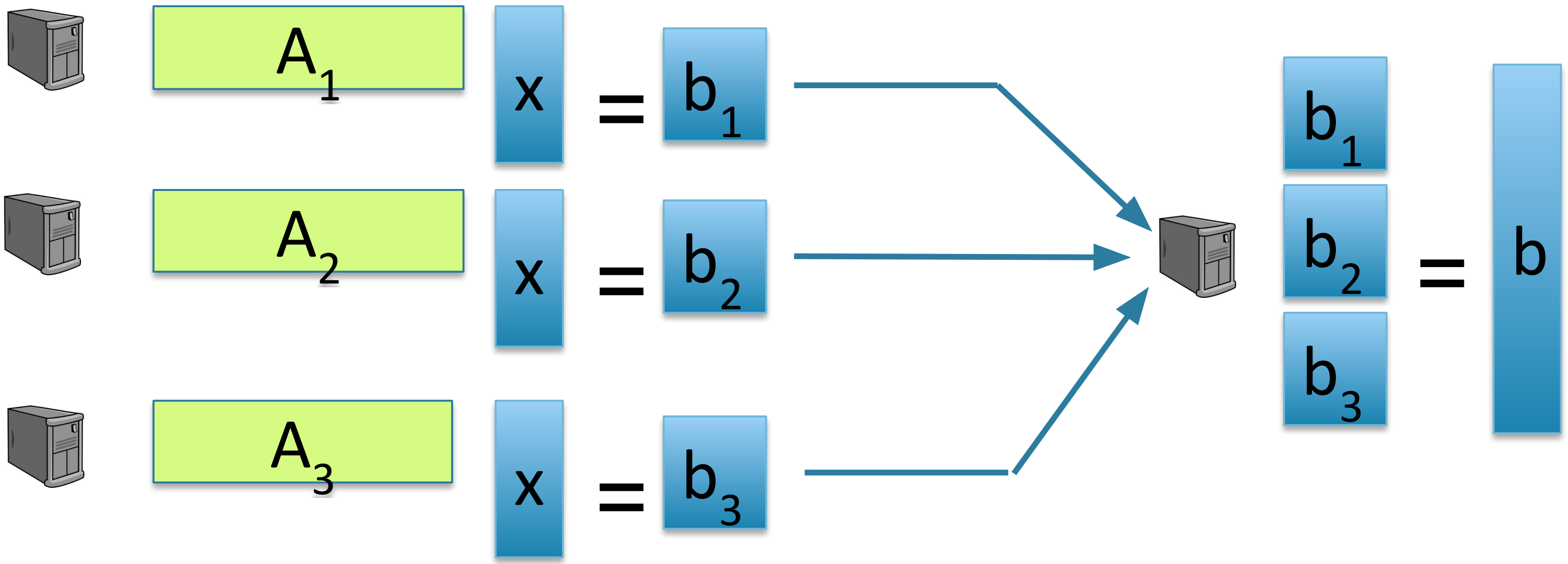
Distributed Matrix Vector Multiplication

- Large Matrices do not fit in memory on a single machine
- Typically stored in a distributed fashion



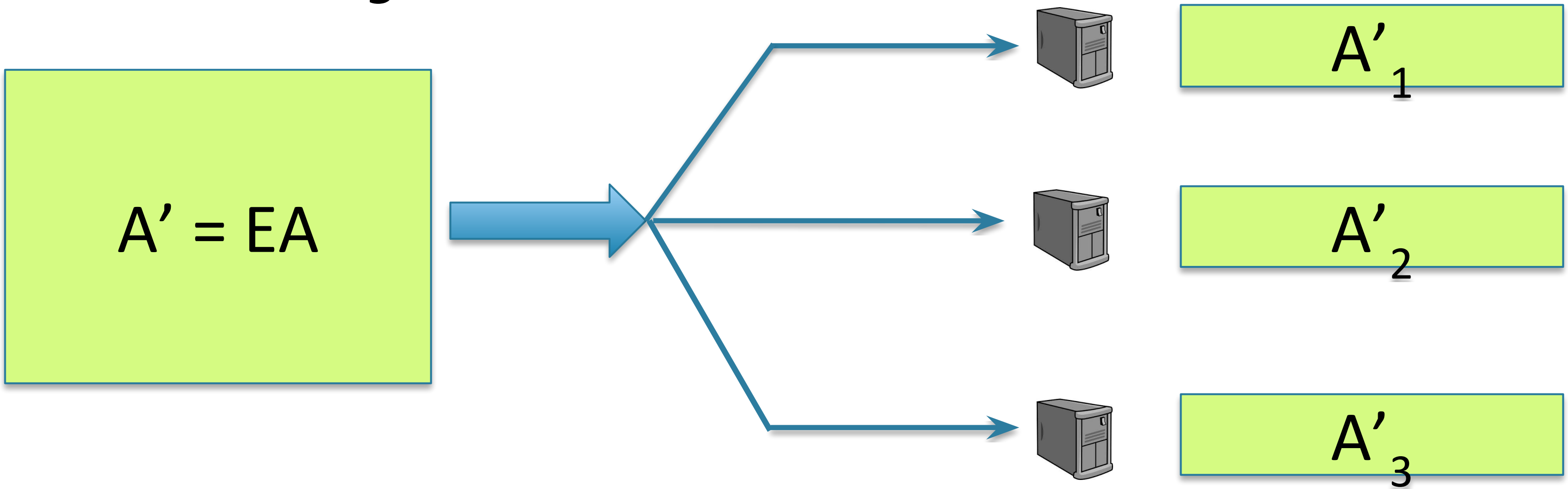
Distributed Matrix Vector Multiplication

- Each submatrix is multiplied with a vector and the results are aggregated to obtain the final product



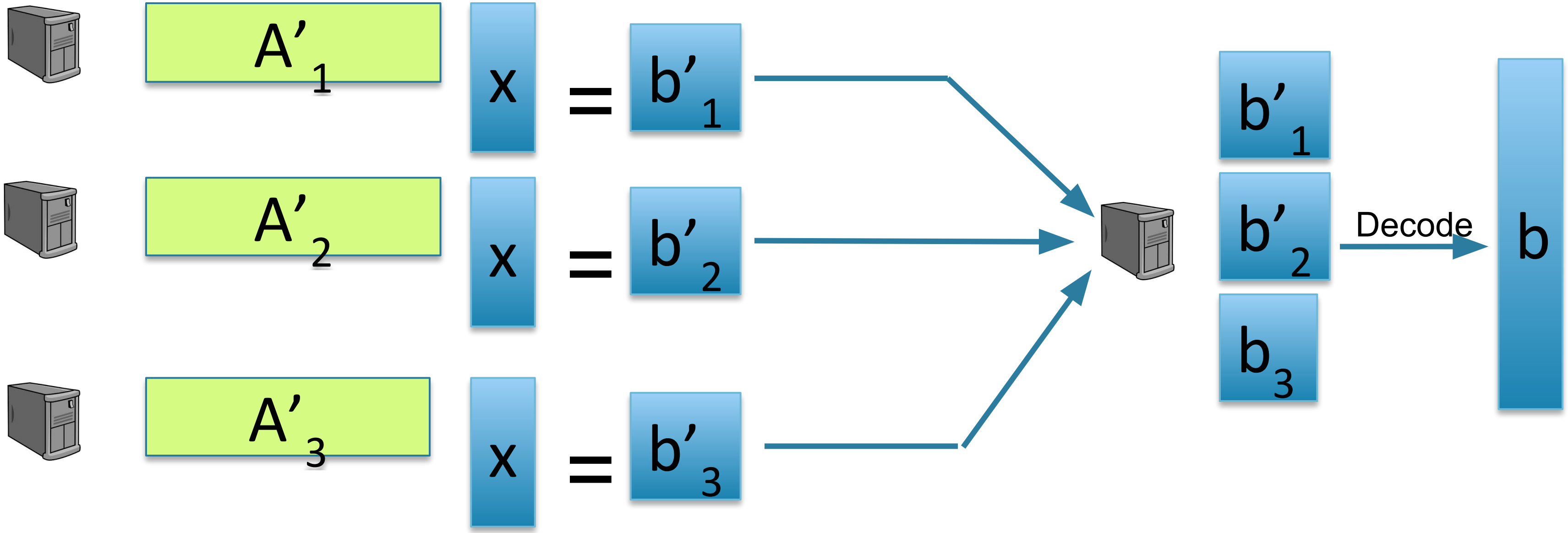
Coded Distributed Matrix Vector Multiplication

- Matrix is encoded by pre-multiplying with a generator matrix before storage



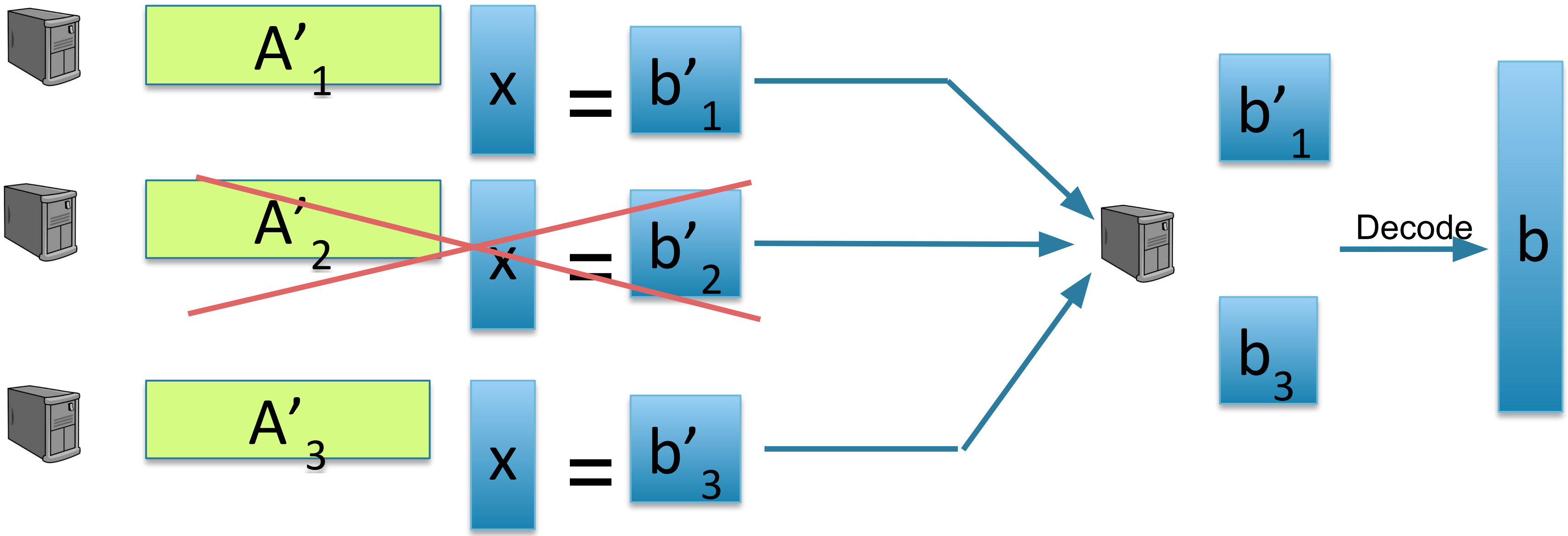
Coded Distributed Matrix Vector Multiplication

- Result of matrix-vector multiplication needs to be decoded to obtain the final product



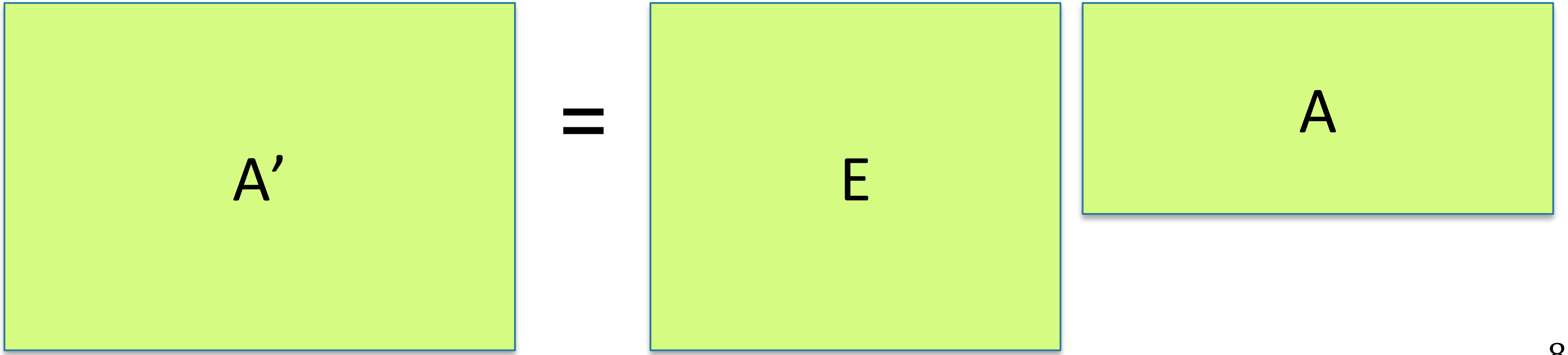
Distributed Matrix Vector Multiplication

- Generator matrix E is chosen so that any 2 of (b'_1,b'_2,b'_3) are sufficient to obtain b



Properties of the Encoding Matrix

- Encoding step: $A' = EA$
  - Size of A = $m \times n$
  - Size of E = $(3m/2) \times m$
  - Size of A′ = $(3m/2) \times n$



Properties of the Encoding Matrix

- If any 2 of (E_1, E_2, E_3) can be aggregated to form an invertible matrix then the matrix vector product Ax can be decoded from any 2 of (A'_2x, A'_3x, A'_3x)

$$\begin{bmatrix} A'_1 \\ A'_2 \\ A'_3 \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} \begin{bmatrix} A \end{bmatrix}$$

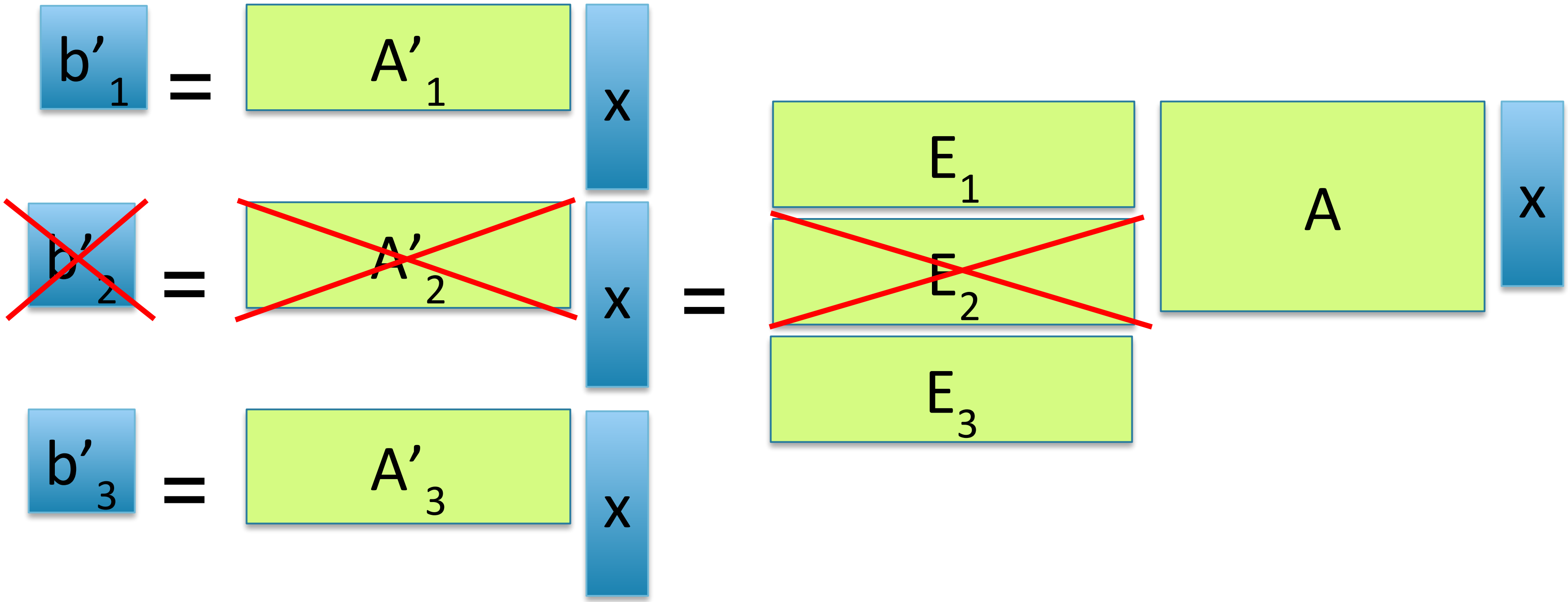
Decoding Process

$$b'_1 = A'_1 \, x$$

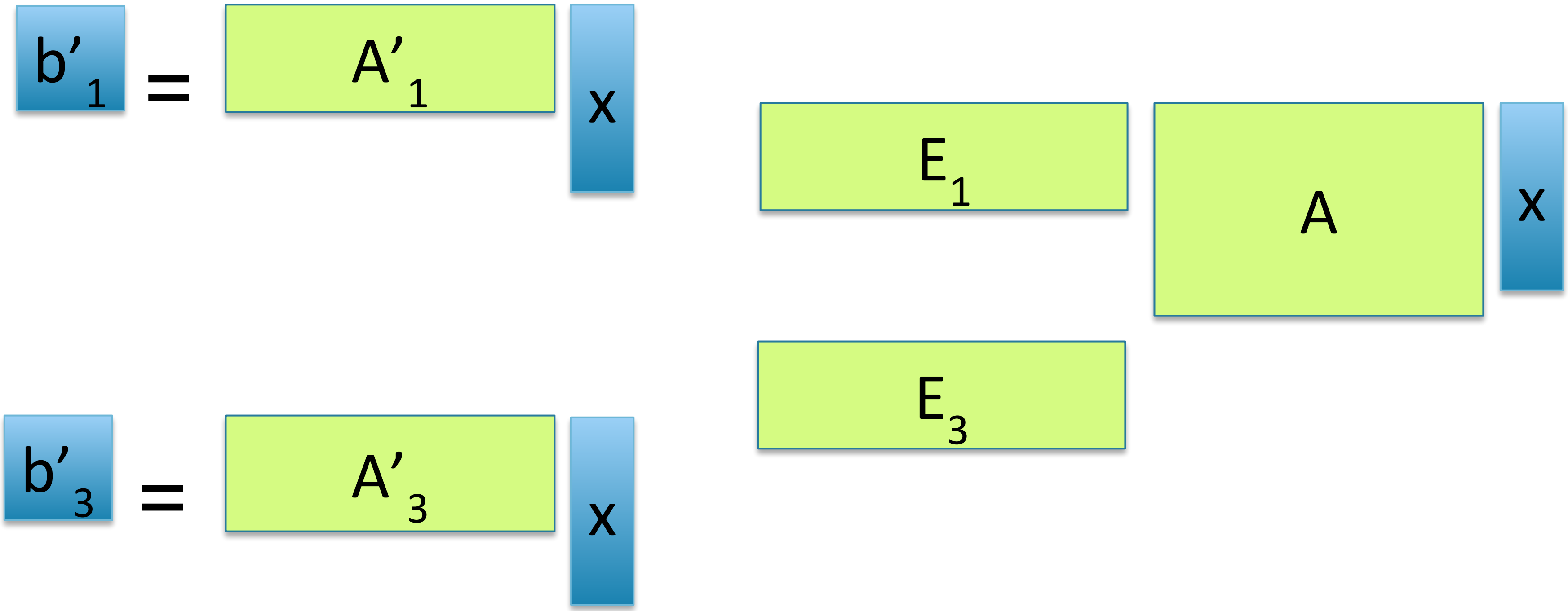
$$b'_2 = A'_2 \, x = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} A \, x$$

$$b'_3 = A'_3 \, x$$

Decoding Process



Decoding Process



Decoding Process

$$b'_1 = A'_1 \; x$$

$$b'_3 = A'_3 \; x = \begin{bmatrix} E_1 \\ E_3 \end{bmatrix} A \; x$$

Decoding Process

$$\begin{bmatrix} b'_1 \\ b'_3 \end{bmatrix} = \begin{bmatrix} E_1 \\ E_3 \end{bmatrix} \begin{bmatrix} A \end{bmatrix} \begin{bmatrix} x \end{bmatrix}$$

Decoding Process

$$\begin{bmatrix} E_1 \\ E_3 \end{bmatrix}^{-1} \begin{matrix} b'_1 \\ b'_3 \end{matrix} = \begin{bmatrix} E_1 \\ E_3 \end{bmatrix}^{-1} \begin{matrix} E_1 \\ E_3 \end{matrix} \; A \; x$$

Decoding Process

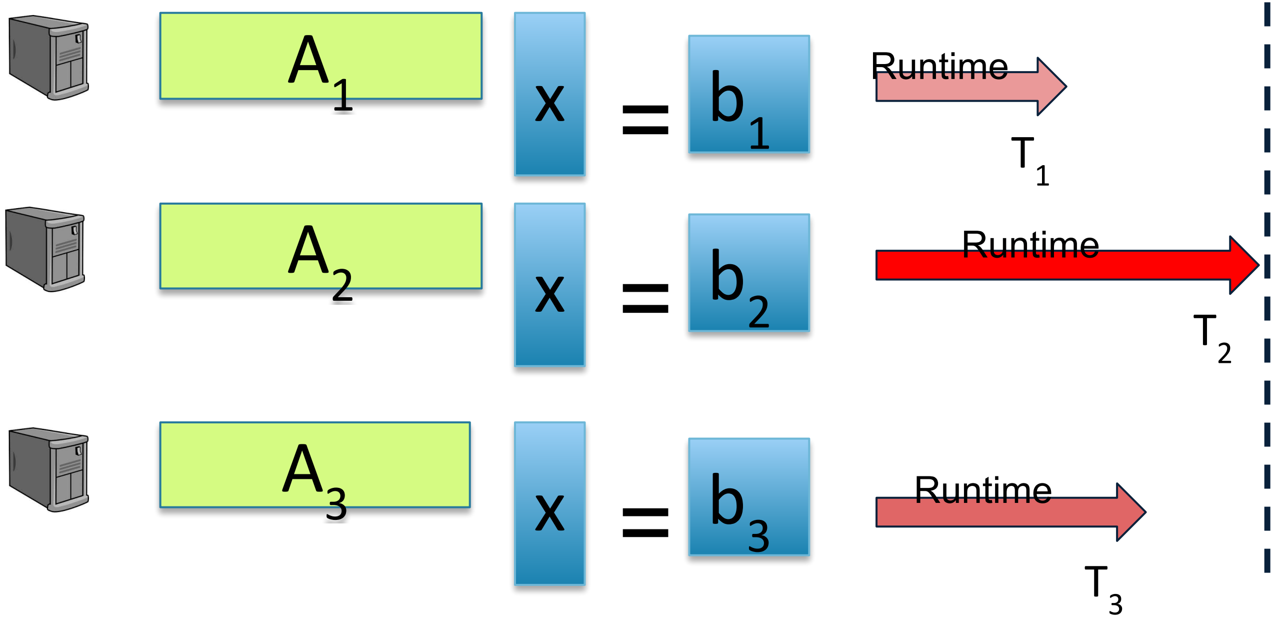
$$\begin{bmatrix} E_1 \\ E_3 \end{bmatrix}^{-1} \begin{bmatrix} b'_1 \\ b'_3 \end{bmatrix} = \cancel{\begin{bmatrix} E_1 \\ E_3 \end{bmatrix}^{-1} \begin{bmatrix} E_1 \\ E_3 \end{bmatrix}} \; A \; x$$

Decoding Process

$$\begin{bmatrix} E_1 \\ E_3 \end{bmatrix}^{-1} \begin{bmatrix} b'_1 \\ b'_3 \end{bmatrix} = A \; x = b$$

Latency Reduction with coding

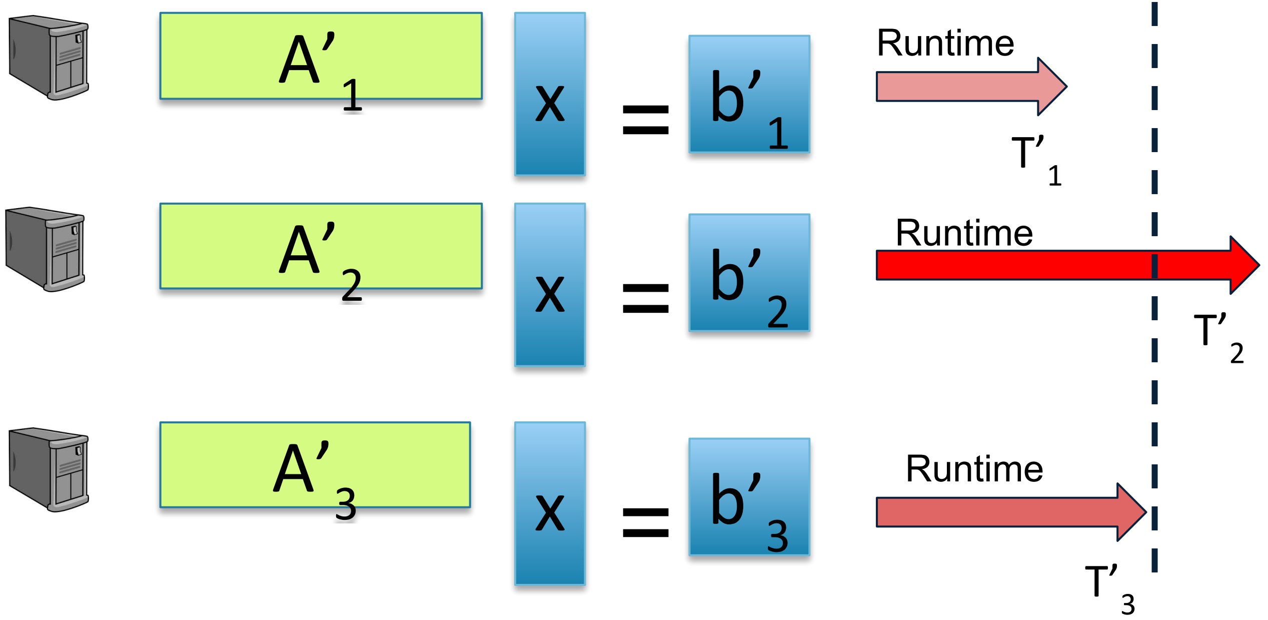
- Without coding we have to wait for all servers to complete their task



Overall Runtime = $T_2 = \max(T_1, T_2, T_3)$

Latency Reduction with coding

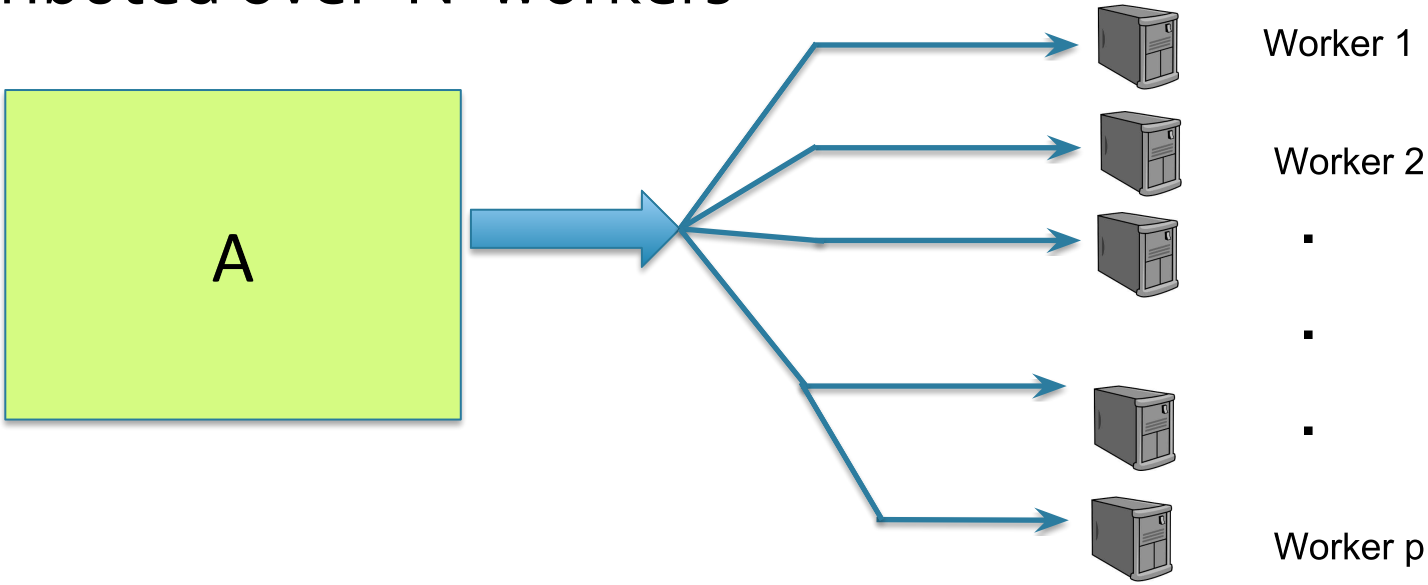
- With coding we only need to wait for the fastest 2 servers



Overall Runtime = $T'_3 < \max(T'_1, T'_2, T'_3)$

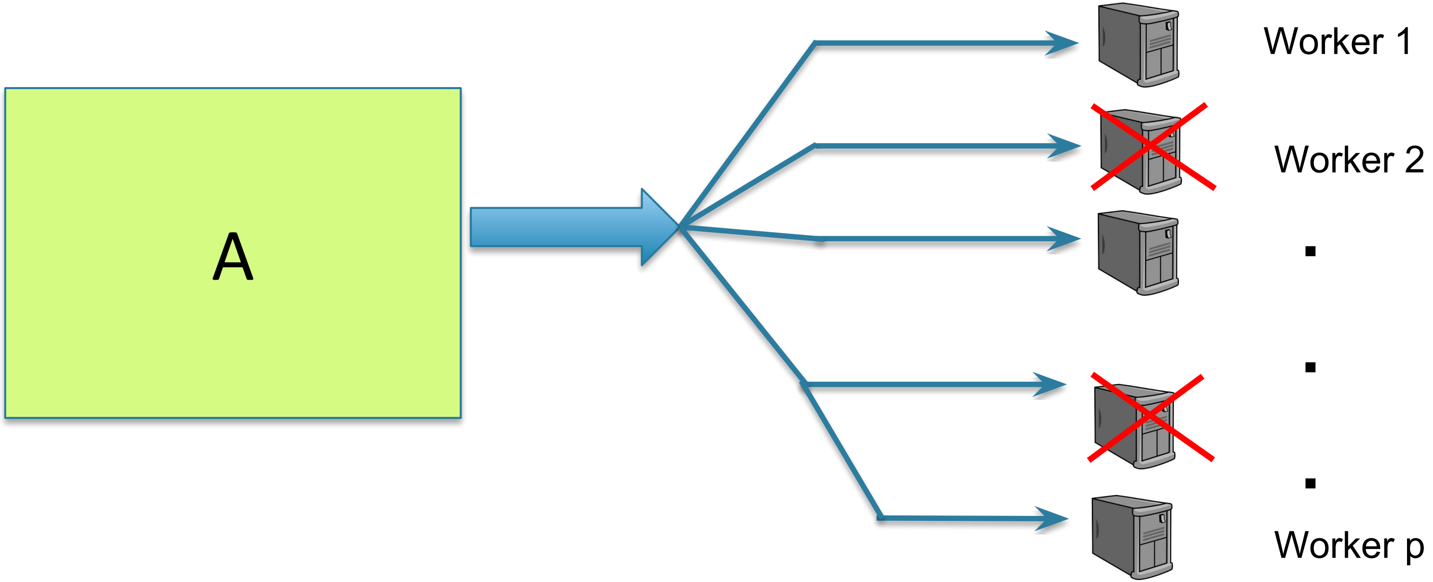
Generalized Coded Computing

- In general the matrix vector multiplication can be distributed over 'N' workers



Generalized Coded Computing

- The goal of coding is to reconstruct the matrix vector product $b = Ax$ from the outputs of any 'k' out of 'N' workers (protects against 'N-k' stragglers)



Generalized Coded Computing

- The encoding scheme consists of splitting matrix A into ‘k’ submatrices and generating N coded symbols using a standard MDS erasure code:

$$\begin{bmatrix} A'_1 \\ A'_2 \\ \vdots \\ A'_k \end{bmatrix} = \mathbf{G} \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{I_k} \\ \mathbf{P} \end{bmatrix}$$

$\mathbf{I_k}$- k x k Identity Matrix

P- Parity check Matrix

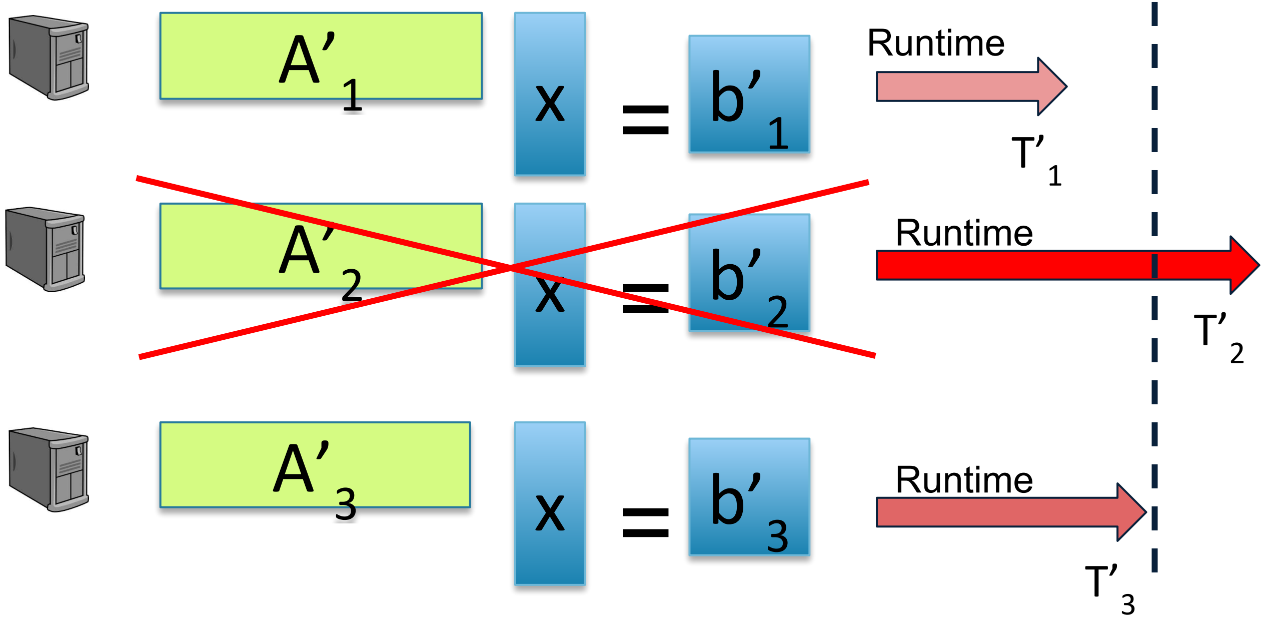
For N = 3, k = 2:

$$\begin{bmatrix} A_1 \\ A_2 \\ A_1+A_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

- Since the encoding scheme is linear, decoding can be achieved using standard MDS decoding from the outputs of any k workers

Drawbacks of the MDS Coded approach

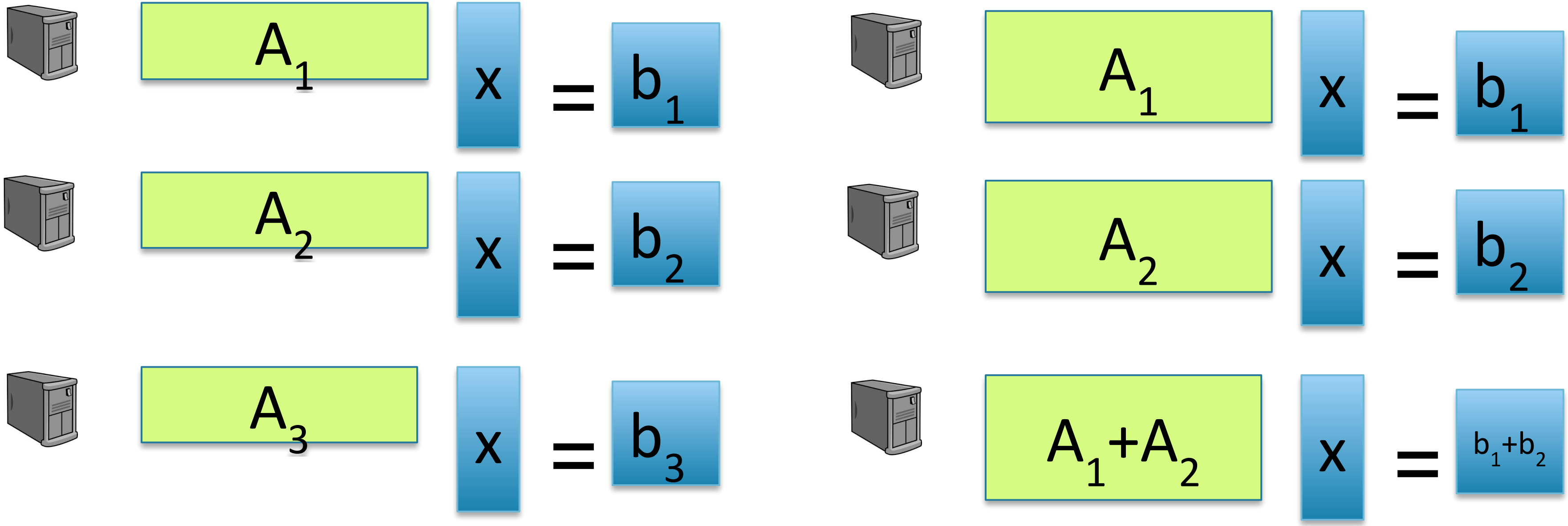
- Neglects partial work done by workers



Overall Runtime = $T'_3 < \max(T'_1, T'_2, T'_3)$

Drawbacks of the MDS Coded approach

- Increases computation load at each individual server



Uncoded

(each worker computes ⅓ of the total task)

MDS-Coded

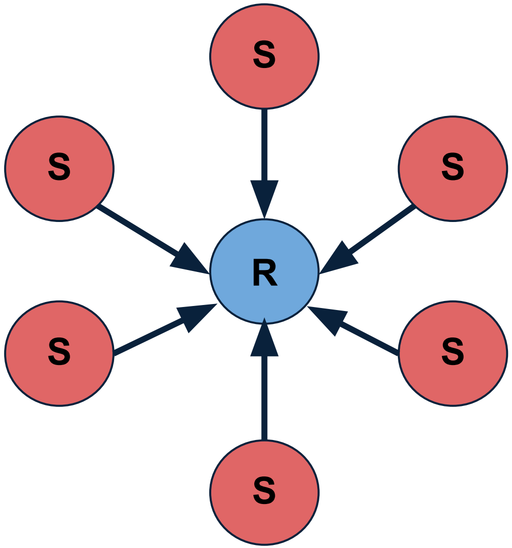
(each worker computes ½ of the total task)

Rateless Erasure Codes

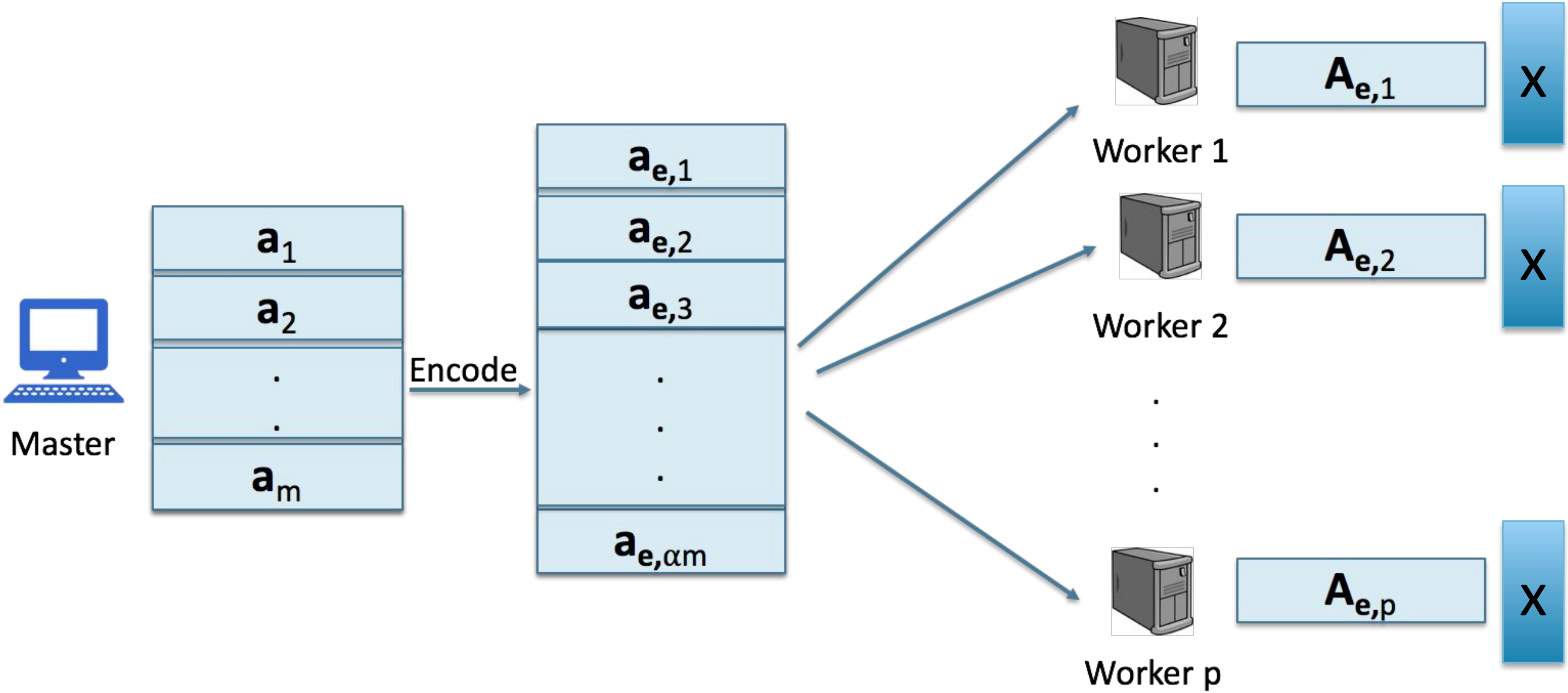
- Erasure codes that can handle a limitless amount of erasures (packet losses)
- Motivated by unreliable communication protocols such as UDP
- Data is communicated at a rapid rate without waiting for acknowledgement from the receiver
- This leads to a high number of packet drops unknown to the sender
- The goal is to reconstruct the original message, with minimal overhead, in the presence of an unbounded number of packet drops, without resending the lost packets
- Rateless Erasure Codes were originally developed by Digital Fountain Inc. (now acquired by Qualcomm) and are used in several wireless communication standards

Mutlipoint-to-Point Transmission

- Waiting for acknowledgements from the receiver leads to time wastage
- If each node communicates the same message then the receiver may receive duplicate messages which is inefficient
- If the message is split across the nodes then erasures lead to loss of data
- Solution: Rateless Erasure Coding (LT/Raptor Codes)



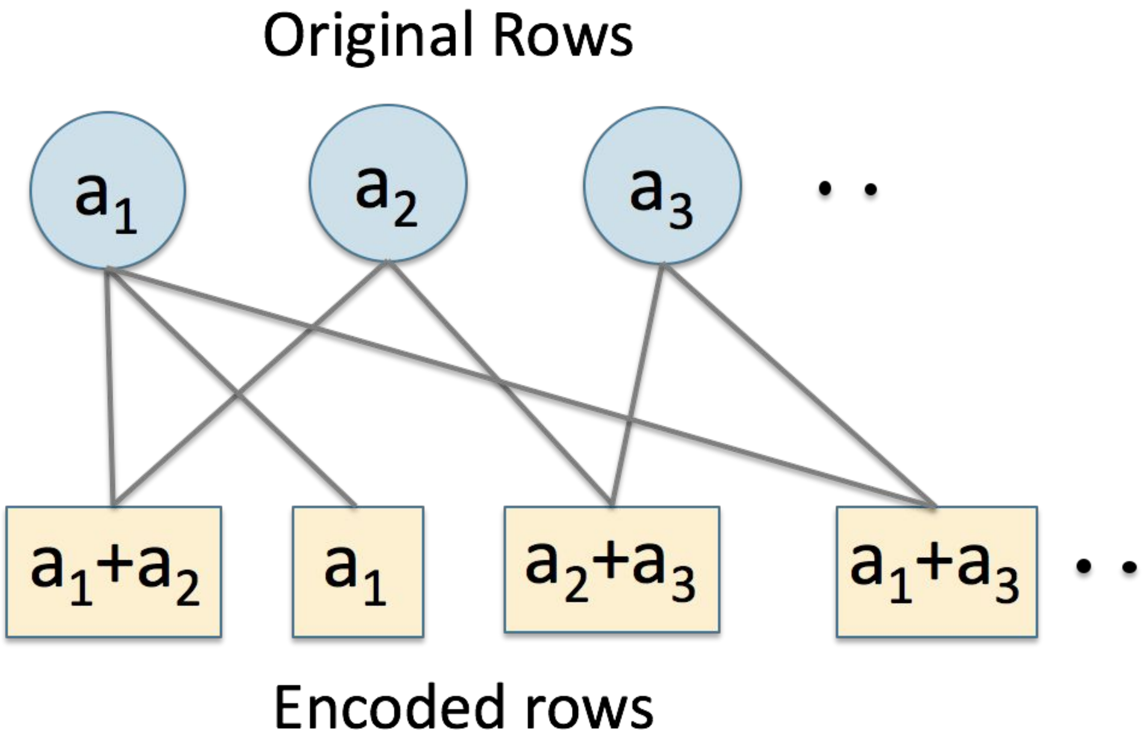
System Model



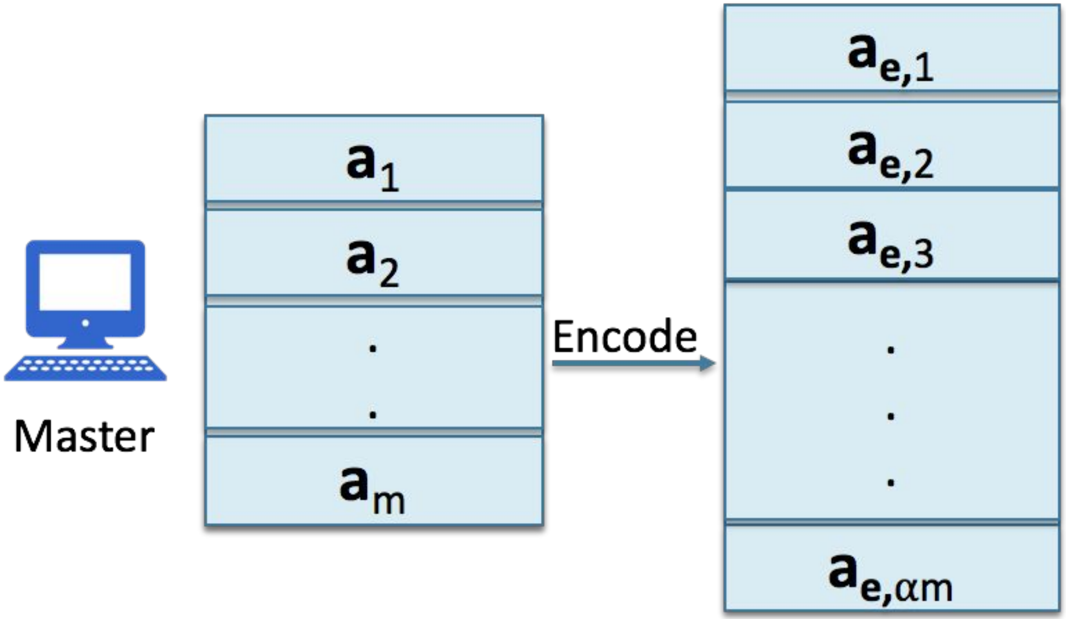
LT Codes (Encoding)

- Determine the degree ‘d’ of an encoding symbol from a given degree distribution ρ(d)
- Choose ‘d’ distinct information symbols uniformly at random
- Generate an encoded symbol which is the sum of the ‘d’ information symbols
- Any number of encoded symbols can be generated

LT Codes (Encoding)



Encoding Computations



- Each encoded matrix row is a linear combination of a random subset of original matrix rows

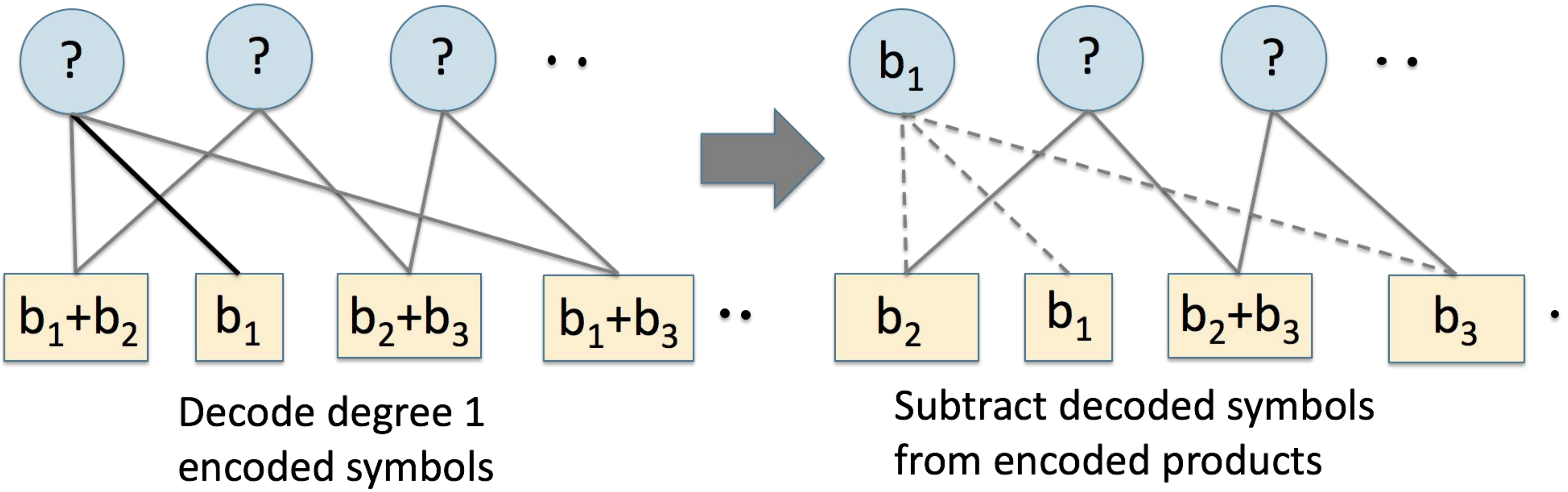
$$\mathbf{a}_{\mathbf{e},j} = \sum_{i \in \mathcal{S}_d} \mathbf{a}_i$$

- The encoded matrix rows are distributed equally across all workers
- We generate 'αm' encoded rows from 'm' original rows (α>1 controls the amount of redundancy)

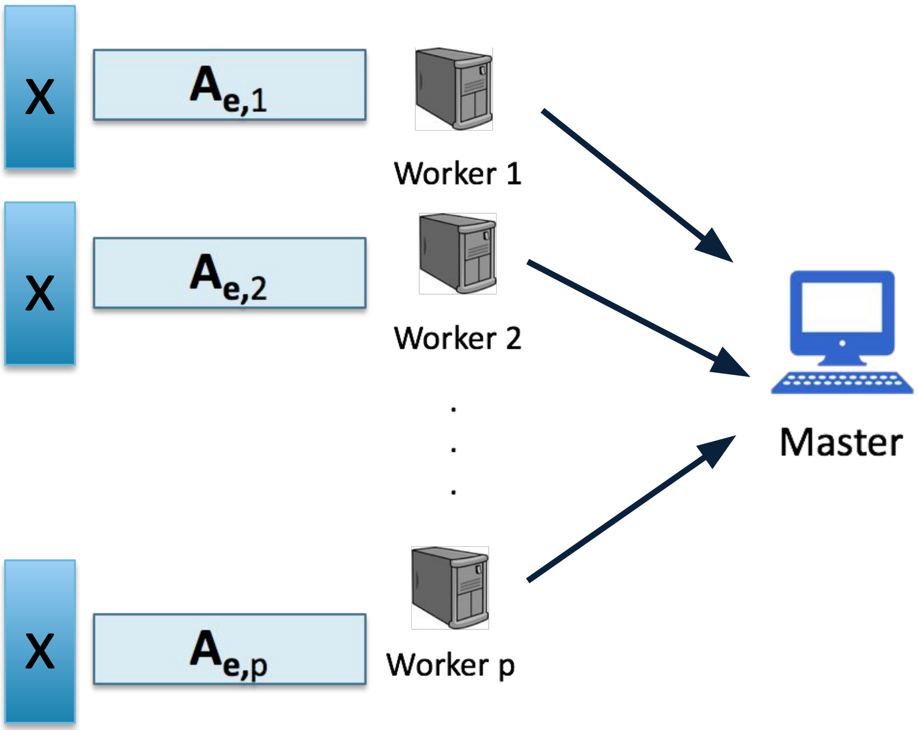
LT Codes (Decoding)

- Identify a symbol with degree 1
- Map that to the corresponding information symbol
- Remove the recovered information symbol from all other encoded symbols containing it
- Repeat until all symbols are successfully decoded

LT Codes (Decoding)



Decoding Computations

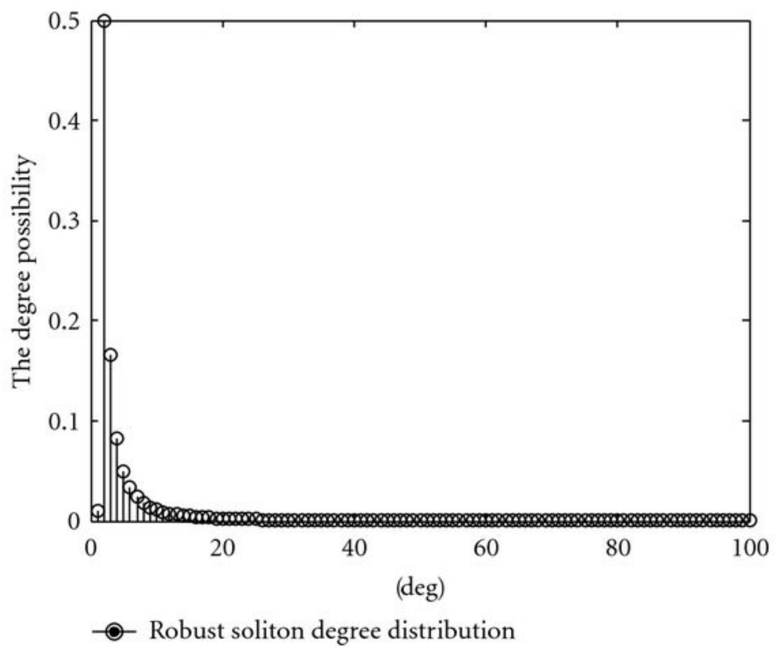


- Workers compute encoded row vector products of the form <$a_{e,j}$,x>
- Master collects a total of m' row vector products from across ***all*** workers (even the slow ones)
- Collected row vector products have the form:

$$< \mathbf{a}_{e,j}, x > = < \sum_{i \in \mathcal{S}_d} \mathbf{a}_i > = \sum_{i \in \mathcal{S}_d} < \mathbf{a}_i, x > = \sum_{i \in \mathcal{S}_d} b_i$$

- LT decoding can be applied to the collected symbols to recover b = $[b_1, b_2, \ldots, b_m]^T$
- Successful decoding occurs with high probability for m' = m(1+ε) where ε -> 0 as m -> ∞

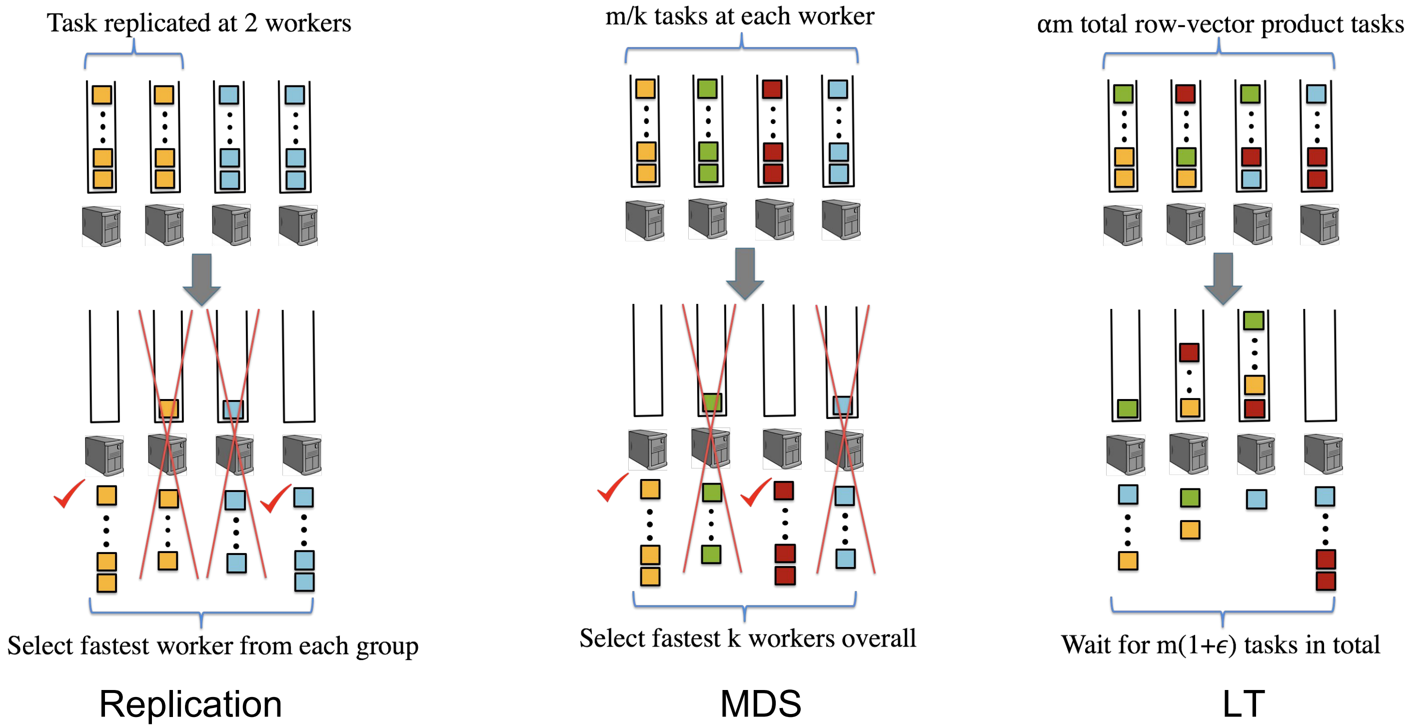
Degree Distribution for LT Codes



The ‘d’ in $\mathbf{a}_{\mathbf{e},j} = \sum_{i \in \mathcal{S}_d} \mathbf{a}_i$ is chosen according to the Robust soliton distribution

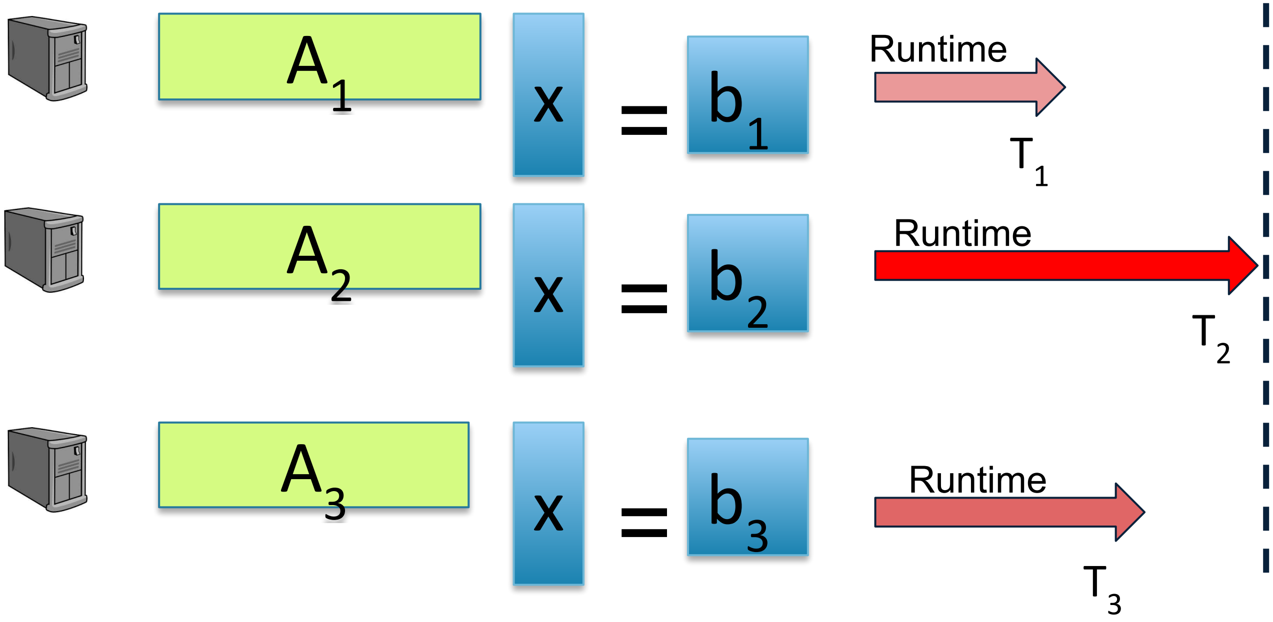
Practical Benefits of using LT Codes

- Partial Work of all workers is Utilised



Practical Benefits of using LT Codes

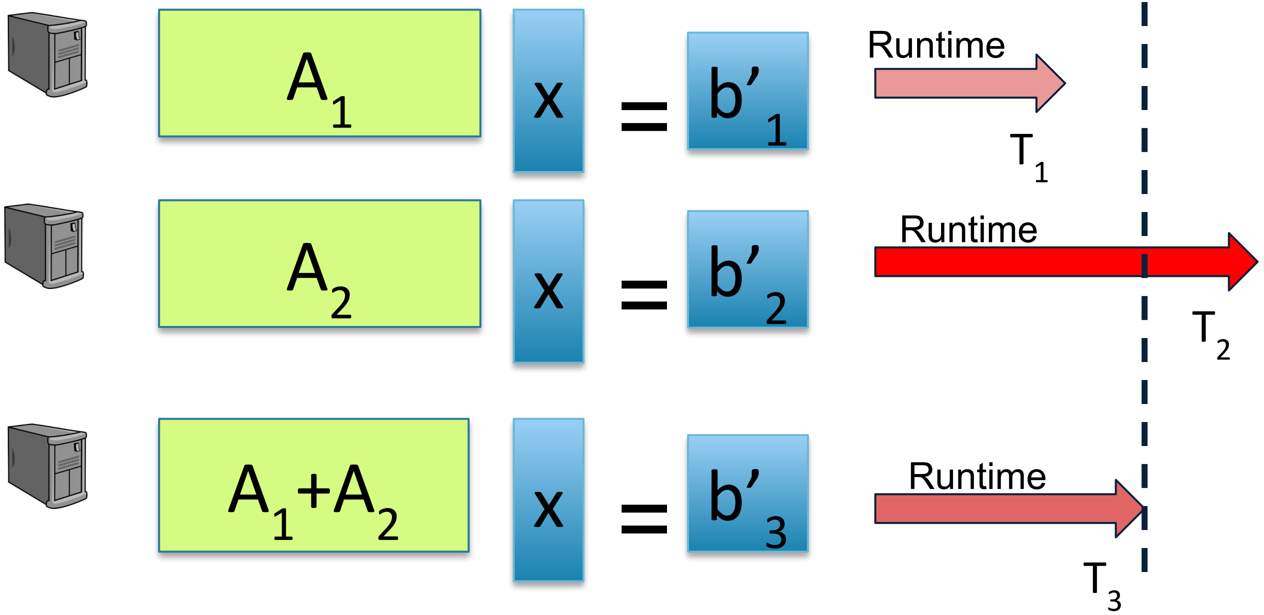
- Reduction in Latency and computation overhead



Overall Runtime for uncoded = T_2 = $\max(T_1, T_2, T_3)$

Practical Benefits of using LT Codes

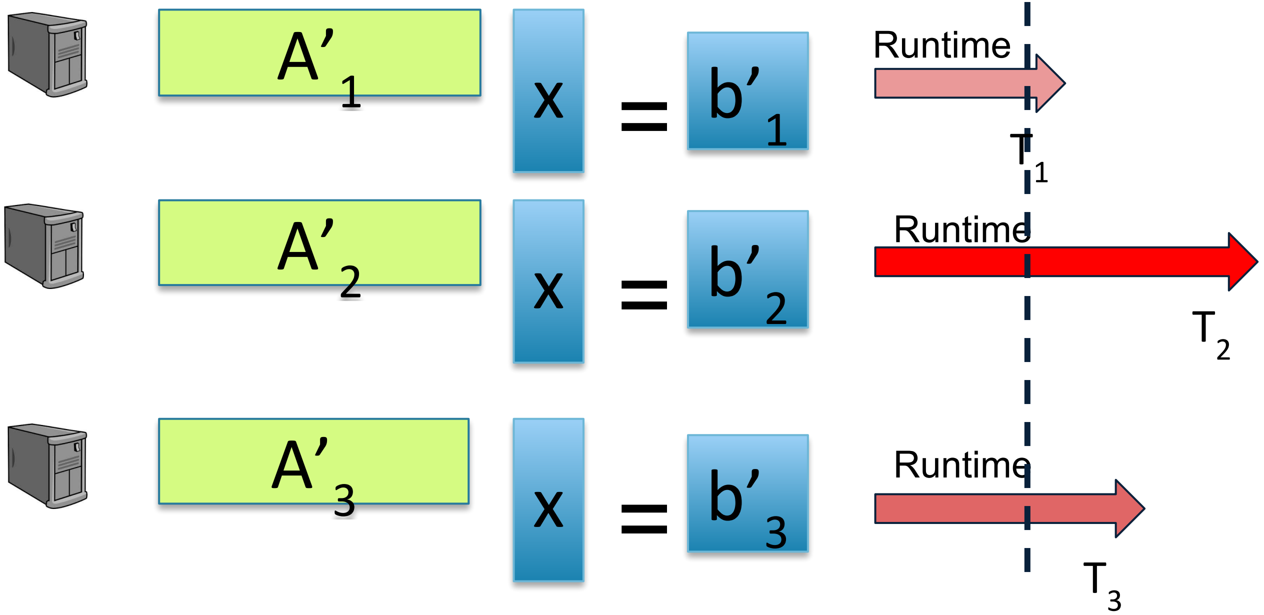
- Reduction in Latency and computation overhead



Overall Runtime for MDS = $T_2 = \max(T_1, T_3)$

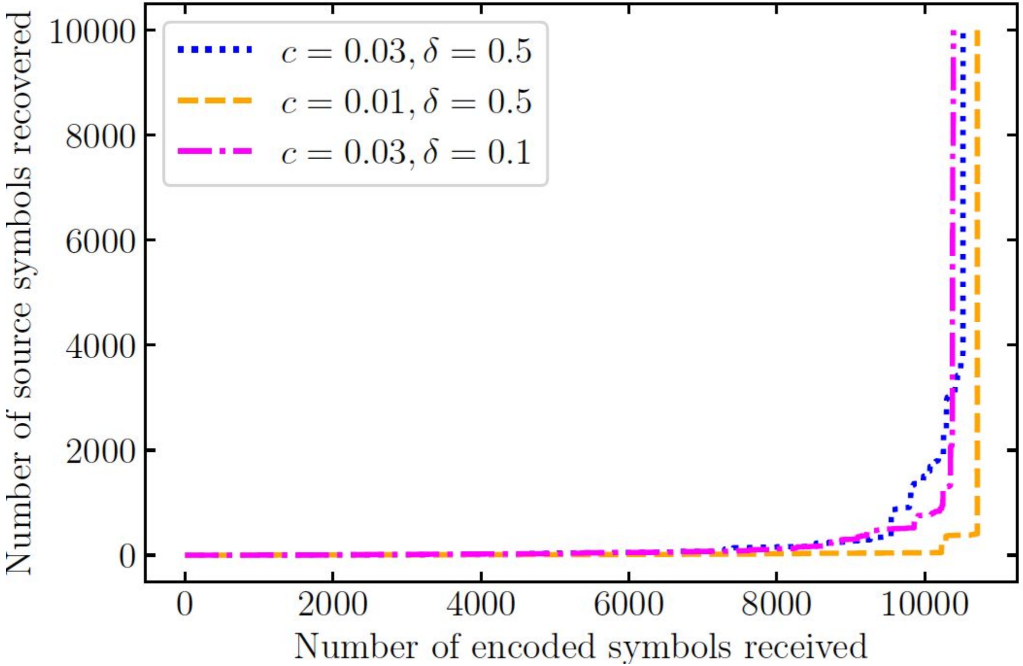
Practical Benefits of using LT Codes

- Reduction in Latency and computation overhead



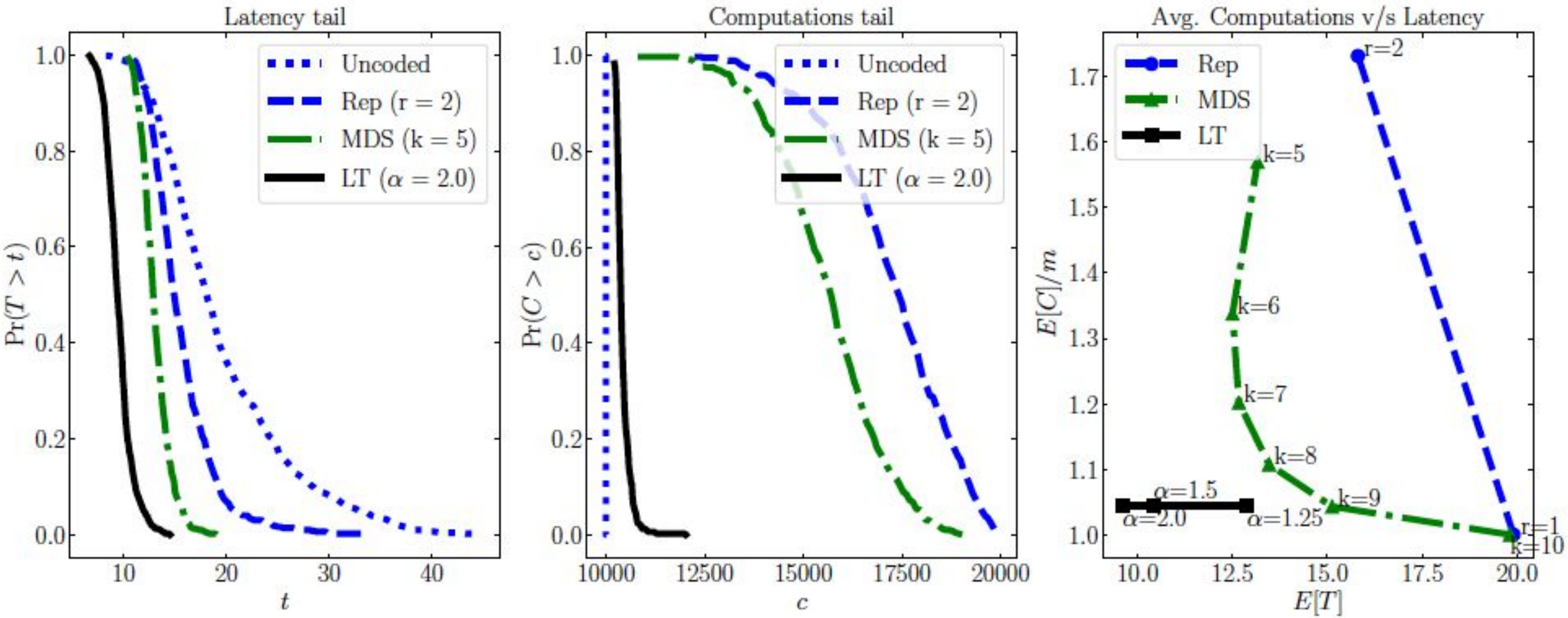
Overall Runtime for LT = T_{LT} = Time taken for all 3 workers to compute a total of m' products of the form $<a_{e,j}, x>$

Simulations



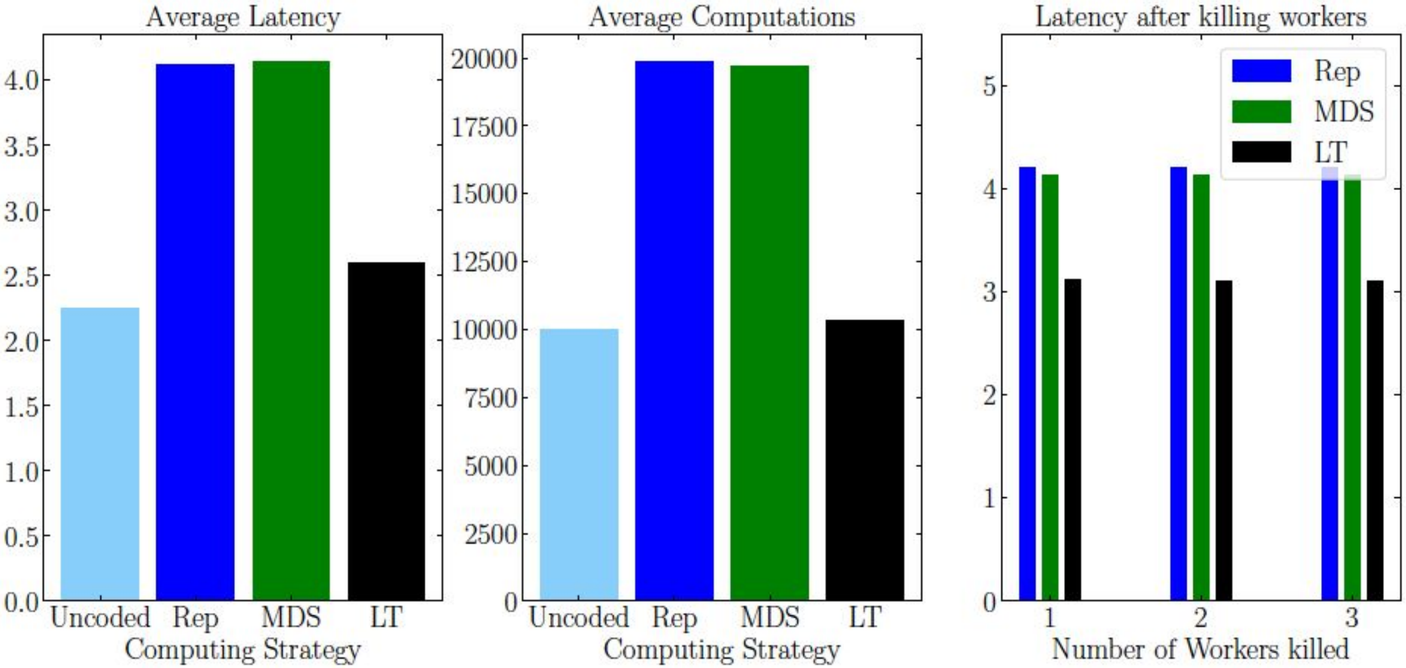
Decoding overhead (difference between m’ and m) for different settings of LT code parameters

Simulations



Simulations are for multiplying a 10000 x 10000 matrix with a 10000 x 1 vector across 10 workers assuming a shifted exponential delay

Experimental Results



Results are for multiplying a 10000 x 10000 matrix with a 10000 x 1 vector across 10 Amazon EC2 workers

Conclusions and Future Work

- Benefits of LT Codes:
  - Efficient utilization of partial work across all workers (both fast and slow)
  - Lower latency and computation overhead at all workers along with better tolerance to worker failures
- Future Directions:
  - Extending to unreliable communication channels between master and workers (erasures/errors in addition to straggling)
  - Extending to other distributed computing tasks beyond matrix-vector multiplication (distributed machine learning)
  - Handling sparsity and other kinds of structure in data (For eg. Low rank matrices)

References

- Mallick, Ankur, Malhar Chaudhari, and Gauri Joshi. "Rateless Codes for Near-Perfect Load Balancing in Distributed Matrix-Vector Multiplication." *arXiv preprint arXiv:1804.10331* (2018) (https://arxiv.org/abs/1804.10331)
- Luby, Michael. "LT codes." IEEE, 2002. (https://www.researchgate.net/profile/Michael_Luby/publication/221498536_LT_codes/links/59274994a6fdcc4443507e45/LT-codes.pdf)