Block-Diagonal Coding for Distributed Computing With Straggling Servers

Albin Severinson[†‡], Alexandre Graell i Amat[†], and Eirik Rosnes[‡]

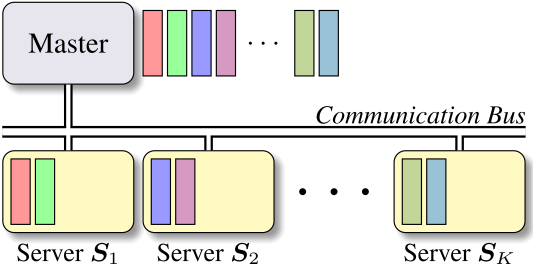
† Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden
‡ University of Bergen/Simula Research Lab, Bergen, Norway

IEEE ITW
Kaohsiung, November, 2017

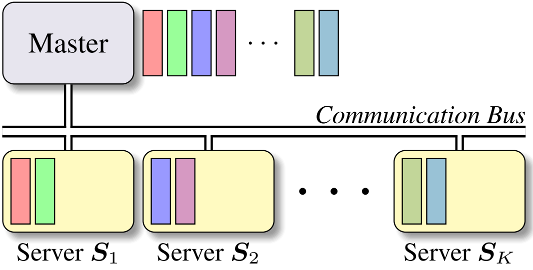
simula@uib

CHALMERS

Motivation



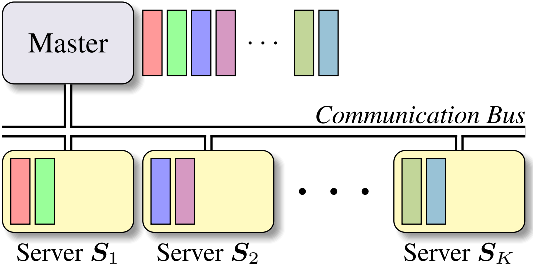
Motivation



Problem addressed

- Given an $m \times n$ matrix $\boldsymbol{A}$ and N vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, we want to compute $\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1, \ldots, \boldsymbol{y}_N = \boldsymbol{A}\boldsymbol{x}_N$ using K servers.

Motivation



Problem addressed

- Given an $m \times n$ matrix $\boldsymbol{A}$ and N vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, we want to compute $\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1, \ldots, \boldsymbol{y}_N = \boldsymbol{A}\boldsymbol{x}_N$ using K servers.

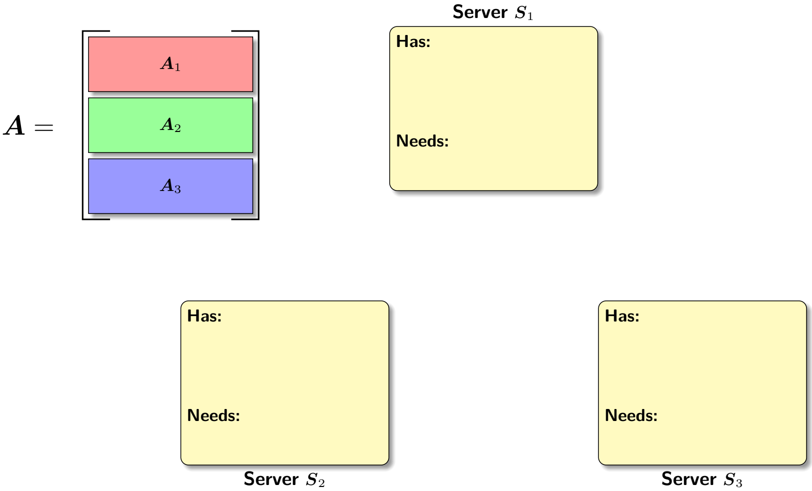
Performance metrics

- Communication load: Average number of messages sent over the network
- Computational delay: Average overall runtime of the computation

Bandwidth Scarcity

(Coded MapReduce, Li *et al.*, 2015)

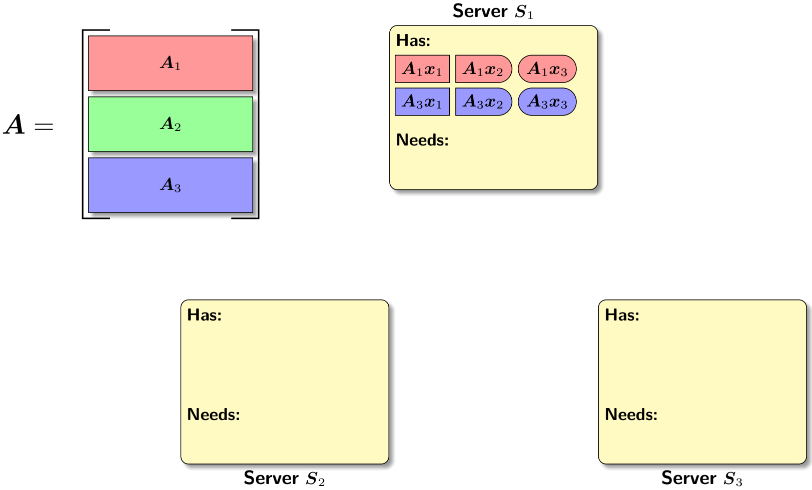
$$\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1, \boldsymbol{y}_2 = \boldsymbol{A}\boldsymbol{x}_2, \boldsymbol{y}_3 = \boldsymbol{A}\boldsymbol{x}_3$$



Bandwidth Scarcity

(Coded MapReduce, Li *et al.*, 2015)

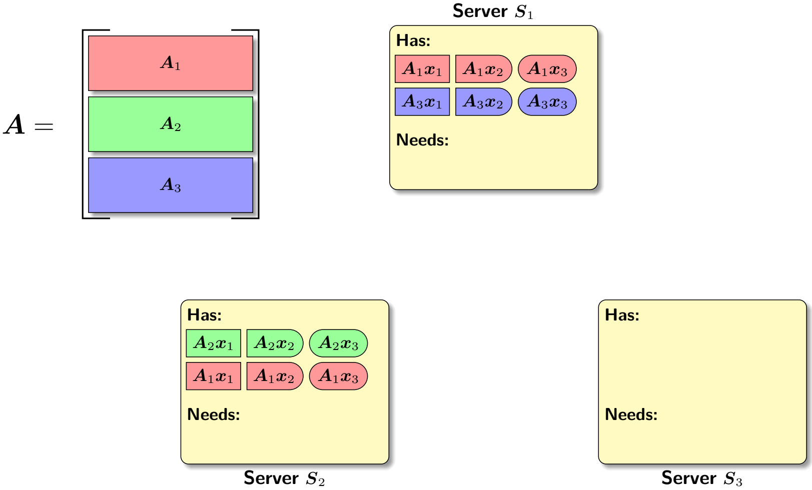
$$y_1 = Ax_1, y_2 = Ax_2, y_3 = Ax_3$$



Bandwidth Scarcity

(Coded MapReduce, Li *et al.*, 2015)

$$\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1, \boldsymbol{y}_2 = \boldsymbol{A}\boldsymbol{x}_2, \boldsymbol{y}_3 = \boldsymbol{A}\boldsymbol{x}_3$$



Bandwidth Scarcity

(Coded MapReduce, Li *et al.*, 2015)

$$\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1, \boldsymbol{y}_2 = \boldsymbol{A}\boldsymbol{x}_2, \boldsymbol{y}_3 = \boldsymbol{A}\boldsymbol{x}_3$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \end{bmatrix}$$

Server S_1

Has:

$\boldsymbol{A}_1\boldsymbol{x}_1$ $\boldsymbol{A}_1\boldsymbol{x}_2$ $\boldsymbol{A}_1\boldsymbol{x}_3$

$\boldsymbol{A}_3\boldsymbol{x}_1$ $\boldsymbol{A}_3\boldsymbol{x}_2$ $\boldsymbol{A}_3\boldsymbol{x}_3$

Needs:

Has:

$\boldsymbol{A}_2\boldsymbol{x}_1$ $\boldsymbol{A}_2\boldsymbol{x}_2$ $\boldsymbol{A}_2\boldsymbol{x}_3$

$\boldsymbol{A}_1\boldsymbol{x}_1$ $\boldsymbol{A}_1\boldsymbol{x}_2$ $\boldsymbol{A}_1\boldsymbol{x}_3$

Needs:

Server S_2

Has:

$\boldsymbol{A}_3\boldsymbol{x}_1$ $\boldsymbol{A}_3\boldsymbol{x}_2$ $\boldsymbol{A}_3\boldsymbol{x}_3$

$\boldsymbol{A}_2\boldsymbol{x}_1$ $\boldsymbol{A}_2\boldsymbol{x}_2$ $\boldsymbol{A}_2\boldsymbol{x}_3$

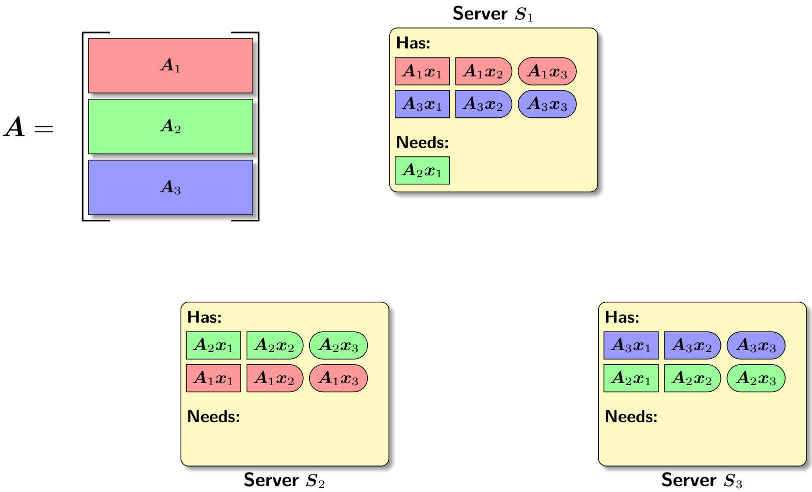
Needs:

Server S_3

Bandwidth Scarcity

(Coded MapReduce, Li *et al.*, 2015)

$$\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1, \boldsymbol{y}_2 = \boldsymbol{A}\boldsymbol{x}_2, \boldsymbol{y}_3 = \boldsymbol{A}\boldsymbol{x}_3$$



Bandwidth Scarcity

(Coded MapReduce, Li *et al.*, 2015)

$$\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1, \boldsymbol{y}_2 = \boldsymbol{A}\boldsymbol{x}_2, \boldsymbol{y}_3 = \boldsymbol{A}\boldsymbol{x}_3$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \end{bmatrix}$$

Server S_1

Has:
$\boldsymbol{A}_1\boldsymbol{x}_1$ $\boldsymbol{A}_1\boldsymbol{x}_2$ $\boldsymbol{A}_1\boldsymbol{x}_3$
$\boldsymbol{A}_3\boldsymbol{x}_1$ $\boldsymbol{A}_3\boldsymbol{x}_2$ $\boldsymbol{A}_3\boldsymbol{x}_3$

Needs:
$\boldsymbol{A}_2\boldsymbol{x}_1$

Has:
$\boldsymbol{A}_2\boldsymbol{x}_1$ $\boldsymbol{A}_2\boldsymbol{x}_2$ $\boldsymbol{A}_2\boldsymbol{x}_3$
$\boldsymbol{A}_1\boldsymbol{x}_1$ $\boldsymbol{A}_1\boldsymbol{x}_2$ $\boldsymbol{A}_1\boldsymbol{x}_3$

Needs:
$\boldsymbol{A}_3\boldsymbol{x}_2$

Server S_2

Has:
$\boldsymbol{A}_3\boldsymbol{x}_1$ $\boldsymbol{A}_3\boldsymbol{x}_2$ $\boldsymbol{A}_3\boldsymbol{x}_3$
$\boldsymbol{A}_2\boldsymbol{x}_1$ $\boldsymbol{A}_2\boldsymbol{x}_2$ $\boldsymbol{A}_2\boldsymbol{x}_3$

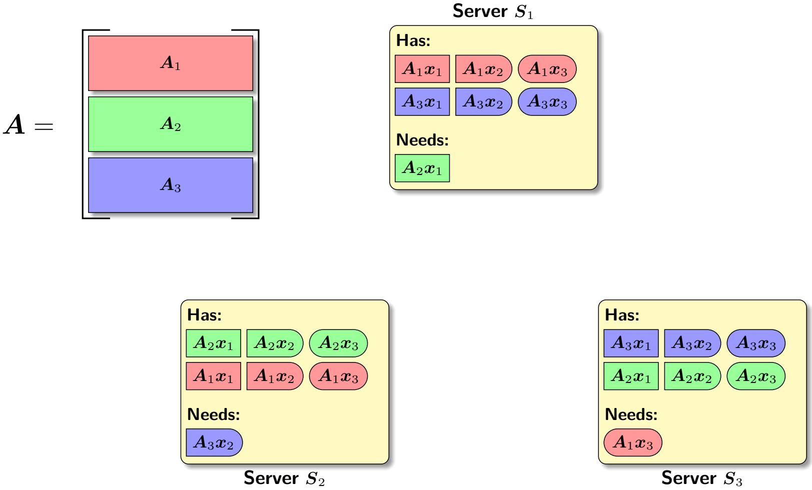
Needs:

Server S_3

Bandwidth Scarcity

(Coded MapReduce, Li *et al.*, 2015)

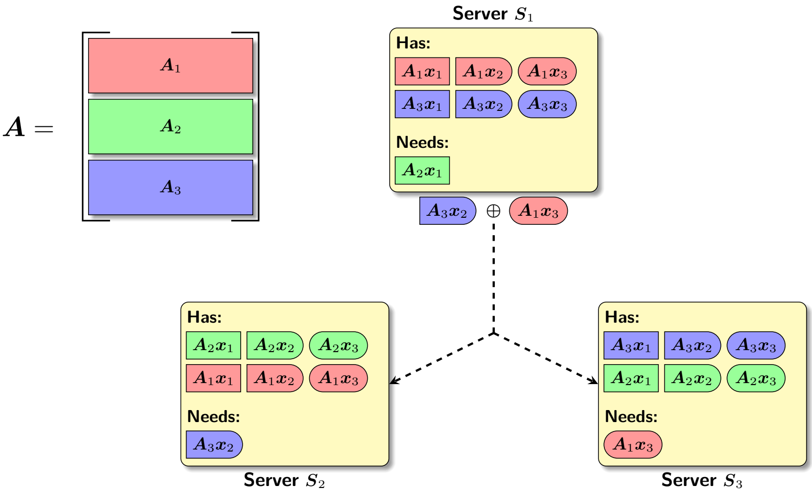
$$\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1, \boldsymbol{y}_2 = \boldsymbol{A}\boldsymbol{x}_2, \boldsymbol{y}_3 = \boldsymbol{A}\boldsymbol{x}_3$$



Bandwidth Scarcity

(Coded MapReduce, Li *et al.*, 2015)

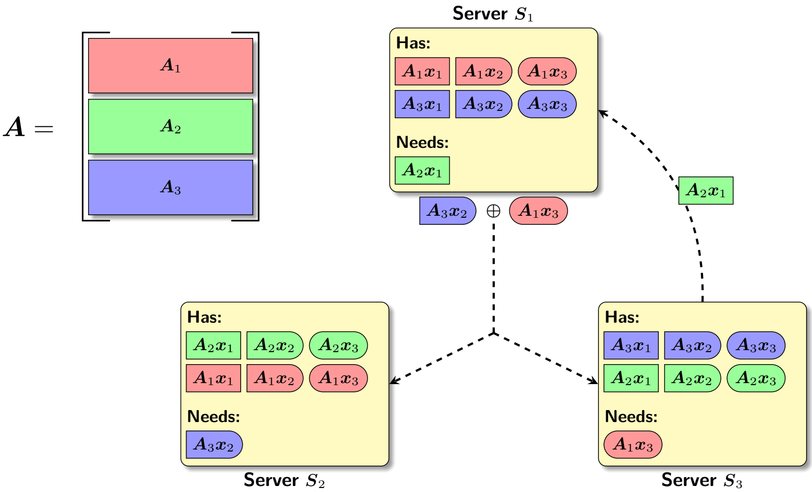
$$\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1, \boldsymbol{y}_2 = \boldsymbol{A}\boldsymbol{x}_2, \boldsymbol{y}_3 = \boldsymbol{A}\boldsymbol{x}_3$$



Bandwidth Scarcity

(Coded MapReduce, Li *et al.*, 2015)

$$\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1, \boldsymbol{y}_2 = \boldsymbol{A}\boldsymbol{x}_2, \boldsymbol{y}_3 = \boldsymbol{A}\boldsymbol{x}_3$$

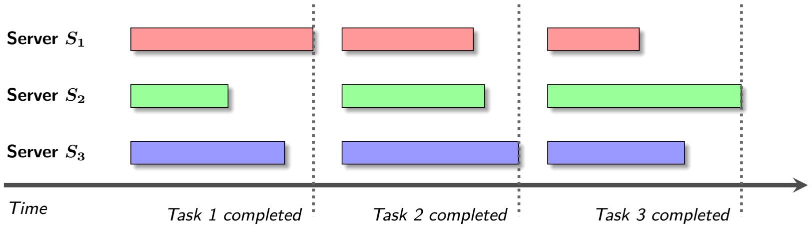


The straggler problem

(Speeding up Distributed Machine Learning Using Codes, Lee *et al.*, 2016)

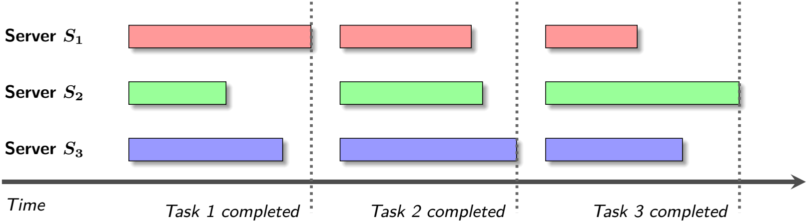
The straggler problem

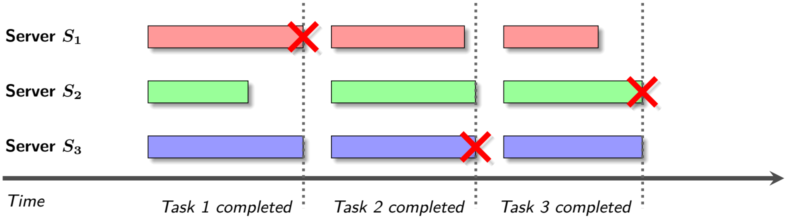
(Speeding up Distributed Machine Learning Using Codes, Lee *et al.*, 2016)



The straggler problem

(Speeding up Distributed Machine Learning Using Codes, Lee *et al.*, 2016)





The Straggler Problem

$$y = Ax$$

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

The Straggler Problem

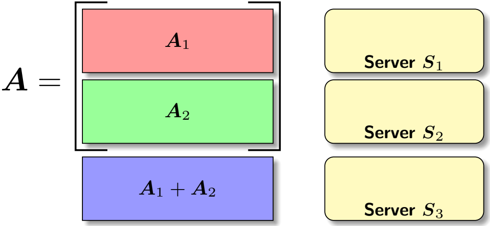
$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \end{bmatrix}$$

$$\boldsymbol{A}_1 + \boldsymbol{A}_2$$

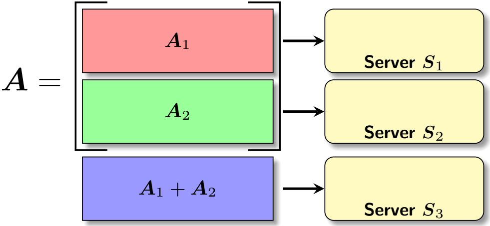
The Straggler Problem

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$



The Straggler Problem

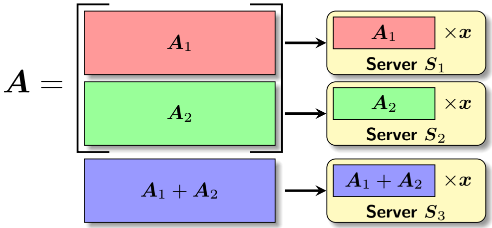
$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$



simula@uib

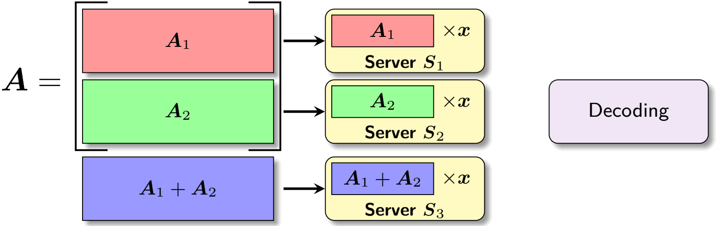
The Straggler Problem

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$



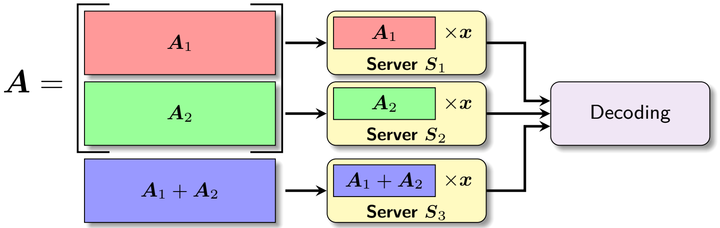
The Straggler Problem

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$



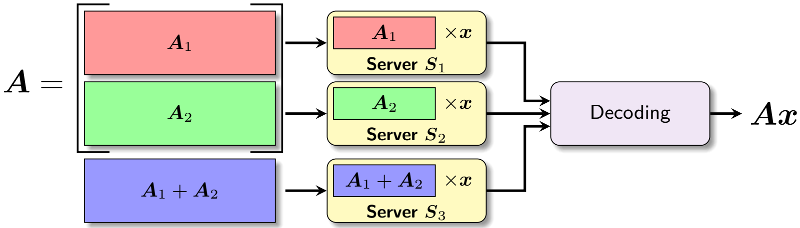
The Straggler Problem

$$y = Ax$$



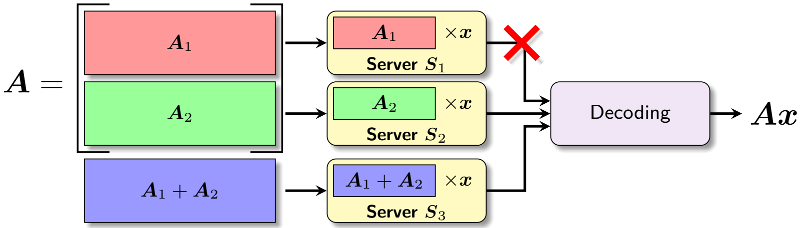
The Straggler Problem

$$y = Ax$$



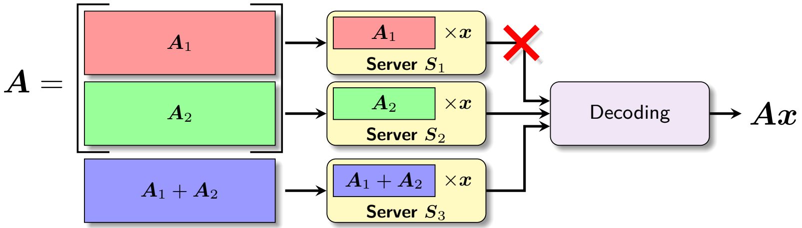
The Straggler Problem

$$y = Ax$$



The Straggler Problem

$$y = Ax$$

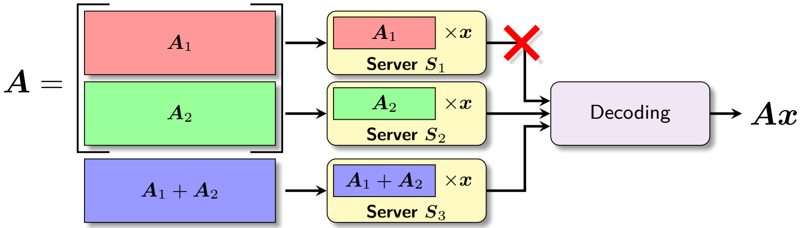


In general

- Introduce redundancy by encoding the input matrix A.

The Straggler Problem

$$y = Ax$$



In general

- Introduce redundancy by encoding the input matrix A.
- Each server is given more work. However, this may still lower the computational delay!

The Unified Coded Framework

(A Unified Coding Framework for Distributed Computing with Straggling Servers, Li *et al.*, 2016)

Coded approach by Li *et al.*

The Unified Coded Framework

(A Unified Coding Framework for Distributed Computing with Straggling Servers, Li *et al.*, 2016)

Coded approach by Li *et al.*

- Encode the columns of $\boldsymbol{A} \in \mathbb{F}^{m \times n}$ using an (r, m) MDS code by multiplying $\boldsymbol{A}$ with an $r \times n$ encoding matrix $\boldsymbol{\Psi}_{\text{MDS}}$, i.e., $\boldsymbol{C} = \boldsymbol{\Psi}_{\text{MDS}} \boldsymbol{A}$.

The Unified Coded Framework

(A Unified Coding Framework for Distributed Computing with Straggling Servers, Li *et al.*, 2016)

Coded approach by Li *et al.*

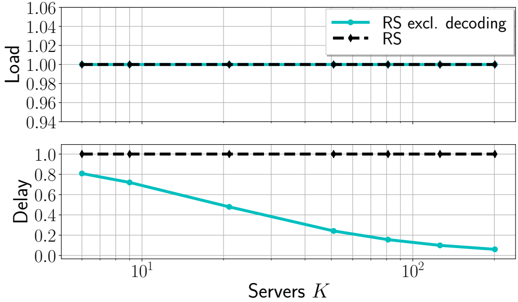
- Encode the columns of $\boldsymbol{A} \in \mathbb{F}^{m \times n}$ using an (r, m) MDS code by multiplying $\boldsymbol{A}$ with an $r \times n$ encoding matrix $\boldsymbol{\Psi}_{\mathsf{MDS}}$, i.e., $\boldsymbol{C} = \boldsymbol{\Psi}_{\mathsf{MDS}}\boldsymbol{A}$.
- But long MDS codes may be complex to decode $\longrightarrow$ decoding delay!

The Unified Coded Framework

(A Unified Coding Framework for Distributed Computing with Straggling Servers, Li *et al.*, 2016)

Coded approach by Li *et al.*

- Encode the columns of $\boldsymbol{A} \in \mathbb{F}^{m \times n}$ using an (r, m) MDS code by multiplying $\boldsymbol{A}$ with an $r \times n$ encoding matrix $\boldsymbol{\Psi}_{\mathsf{MDS}}$, i.e., $\boldsymbol{C} = \boldsymbol{\Psi}_{\mathsf{MDS}}\boldsymbol{A}$.
- But long MDS codes may be complex to decode $\longrightarrow$ decoding delay!



- 2000 rows assigned to each server, $n = 10000$ columns, $N = 2K/3$ vectors, and code rate $m/r = 2/3$.

In this talk

Block-Diagonal Coding

- We propose a block-diagonal encoding scheme with lower decoding complexity.

In this talk

Block-Diagonal Coding

- We propose a block-diagonal encoding scheme with lower decoding complexity.

Idea

- Partition $\boldsymbol{A}$ into T disjoint submatrices and apply smaller MDS codes to each submatrix,

$$\boldsymbol{C} = \boldsymbol{\Psi}_{\mathsf{BDC}}\boldsymbol{A}, \quad \boldsymbol{\Psi}_{\mathsf{BDC}} = \begin{bmatrix} \boldsymbol{\psi}_1 & & \\ & \ddots & \\ & & \boldsymbol{\psi}_T \end{bmatrix}.$$

In this talk

Block-Diagonal Coding

- We propose a block-diagonal encoding scheme with lower decoding complexity.

Idea

- Partition $\boldsymbol{A}$ into T disjoint submatrices and apply smaller MDS codes to each submatrix,

$$\boldsymbol{C} = \boldsymbol{\Psi}_{\text{BDC}}\boldsymbol{A}, \quad \boldsymbol{\Psi}_{\text{BDC}} = \begin{bmatrix} \psi_1 & & \\ & \ddots & \\ & & \psi_T \end{bmatrix}.$$

Outcome

- No impact on the communication load or the computational delay of the matrix multiplication up to a given T.

In this talk

Block-Diagonal Coding

- We propose a block-diagonal encoding scheme with lower decoding complexity.

Idea

- Partition $\boldsymbol{A}$ into T disjoint submatrices and apply smaller MDS codes to each submatrix,

$$\boldsymbol{C} = \boldsymbol{\Psi}_{\mathsf{BDC}}\boldsymbol{A}, \quad \boldsymbol{\Psi}_{\mathsf{BDC}} = \begin{bmatrix} \psi_1 & & \\ & \ddots & \\ & & \psi_T \end{bmatrix}.$$

Outcome

- No impact on the communication load or the computational delay of the matrix multiplication up to a given T.
- Overall computational delay is lower than that of the scheme by Li *et al.*.

In this talk

Block-Diagonal Coding

- We propose a block-diagonal encoding scheme with lower decoding complexity.

Idea

- Partition $\boldsymbol{A}$ into T disjoint submatrices and apply smaller MDS codes to each submatrix,

$$\boldsymbol{C} = \boldsymbol{\Psi}_{\text{BDC}}\boldsymbol{A}, \quad \boldsymbol{\Psi}_{\text{BDC}} = \begin{bmatrix} \psi_1 & & \\ & \ddots & \\ & & \psi_T \end{bmatrix}.$$

Outcome

- No impact on the communication load or the computational delay of the matrix multiplication up to a given T.
- Overall computational delay is lower than that of the scheme by Li *et al.*.
- Larger T may reduce computational delay further at the expense of higher communication load.

Block-Diagonal Coding

$$\Psi_{\mathsf{BDC}} = \underbrace{\begin{bmatrix} \psi_1 & & \\ & \psi_2 & \\ & & \psi_3 \end{bmatrix}}_{r \times m}$$

- Block-diagonal encoding with $T = 3$ partitions.

Block-Diagonal Coding

$$\Psi_{\mathsf{BDC}} \; \boldsymbol{A} = \underbrace{\begin{bmatrix} \boldsymbol{\psi}_1 & & \\ & \boldsymbol{\psi}_2 & \\ & & \boldsymbol{\psi}_3 \end{bmatrix}}_{r \times m} \underbrace{\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \end{bmatrix}}_{m \times n}$$

- Block-diagonal encoding with $T = 3$ partitions.

Block-Diagonal Coding

$$\boldsymbol{\Psi}_{\text{BDC}}\;\boldsymbol{A} = \underbrace{\begin{bmatrix} \boldsymbol{\psi}_1 & & \\ & \boldsymbol{\psi}_2 & \\ & & \boldsymbol{\psi}_3 \end{bmatrix}}_{r \times m} \underbrace{\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \end{bmatrix}}_{m \times n} = \underbrace{\begin{bmatrix} \phantom{\boldsymbol{A}} \end{bmatrix}}_{r \times n}$$

- Block-diagonal encoding with $T = 3$ partitions.

Block-Diagonal Coding

$$\Psi_{\text{BDC}} \, \boldsymbol{A} = \underbrace{\begin{bmatrix} \boldsymbol{\psi}_1 & & \\ & \boldsymbol{\psi}_2 & \\ & & \boldsymbol{\psi}_3 \end{bmatrix}}_{r \times m} \underbrace{\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \end{bmatrix}}_{m \times n} = \underbrace{\begin{bmatrix} \boldsymbol{\psi}_1 \boldsymbol{A}_1 \\ \\ \\ \end{bmatrix}}_{r \times n}$$

- Block-diagonal encoding with $T = 3$ partitions.

Block-Diagonal Coding

$$\boldsymbol{\Psi}_{\text{BDC}} \; \boldsymbol{A} = \underset{r \times m}{\begin{bmatrix} \boldsymbol{\psi}_1 & & \\ & \boldsymbol{\psi}_2 & \\ & & \boldsymbol{\psi}_3 \end{bmatrix}} \underset{m \times n}{\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \end{bmatrix}} = \underset{r \times n}{\begin{bmatrix} \boldsymbol{\psi}_1 \boldsymbol{A}_1 \\ \boldsymbol{\psi}_2 \boldsymbol{A}_2 \\ \phantom{\boldsymbol{\psi}_3 \boldsymbol{A}_3} \end{bmatrix}}$$

- Block-diagonal encoding with $T = 3$ partitions.

Block-Diagonal Coding

$$\Psi_{\text{BDC}}\ \boldsymbol{A} = \underbrace{\begin{bmatrix} \boldsymbol{\psi}_1 & & \\ & \boldsymbol{\psi}_2 & \\ & & \boldsymbol{\psi}_3 \end{bmatrix}}_{r \times m} \underbrace{\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \end{bmatrix}}_{m \times n} = \underbrace{\begin{bmatrix} \boldsymbol{\psi}_1 \boldsymbol{A}_1 \\ \boldsymbol{\psi}_2 \boldsymbol{A}_2 \\ \boldsymbol{\psi}_3 \boldsymbol{A}_3 \end{bmatrix}}_{r \times n}$$

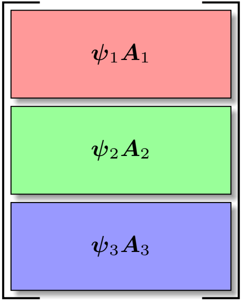
- Block-diagonal encoding with $T = 3$ partitions.

Block-Diagonal Coding

$$\underset{}{\boldsymbol{\Psi}_{\mathsf{BDC}}}\ \boldsymbol{A} = \underbrace{\begin{bmatrix} \boldsymbol{\psi}_1 & & \\ & \boldsymbol{\psi}_2 & \\ & & \boldsymbol{\psi}_3 \end{bmatrix}}_{r \times m} \underbrace{\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \end{bmatrix}}_{m \times n} = \underbrace{\begin{bmatrix} \boldsymbol{\psi}_1\boldsymbol{A}_1 \\ \boldsymbol{\psi}_2\boldsymbol{A}_2 \\ \boldsymbol{\psi}_3\boldsymbol{A}_3 \end{bmatrix}}_{r \times n}$$

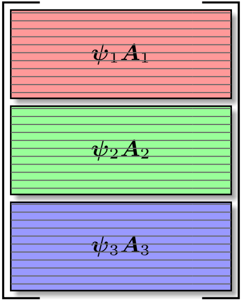
- Block-diagonal encoding with $T = 3$ partitions.
- Need any m/T out of r/T rows from each partition to decode.

Block-Diagonal Coding



- Block-diagonal encoding with $T = 3$ partitions.
- Need any m/T out of r/T rows from each partition to decode.

Block-Diagonal Coding

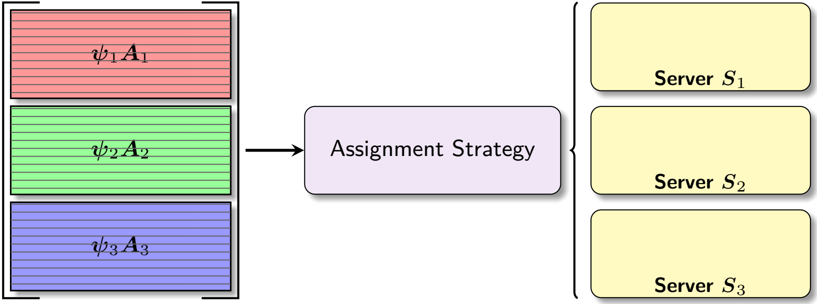


- Block-diagonal encoding with $T = 3$ partitions.
- Need any m/T out of r/T rows from each partition to decode.

Block-Diagonal Coding

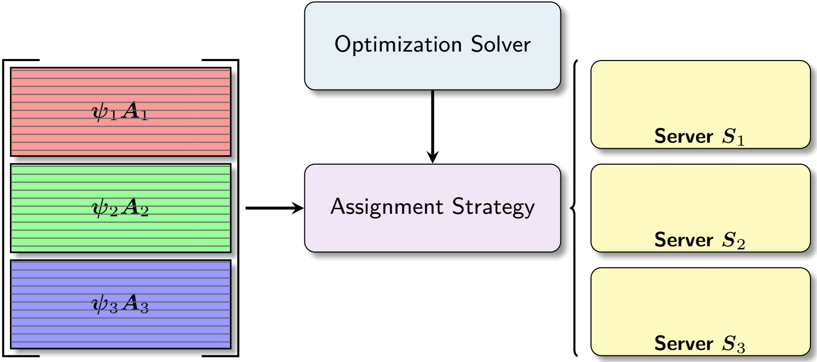
- Block-diagonal encoding with $T = 3$ partitions.
- Need any m/T out of r/T rows from each partition to decode.

Block-Diagonal Coding



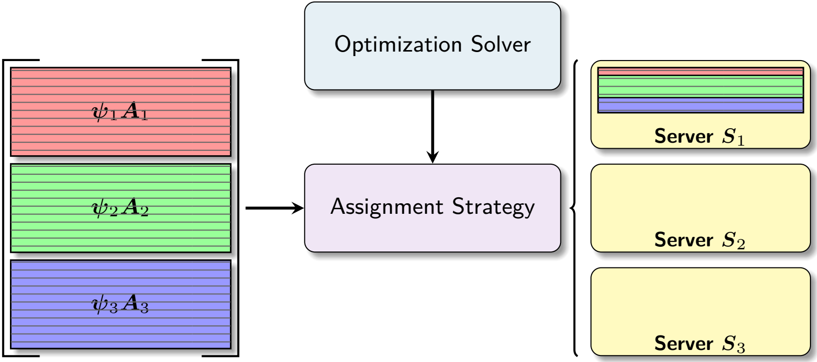
- Block-diagonal encoding with $T = 3$ partitions.
- Need any m/T out of r/T rows from each partition to decode.
- Need to assign coded rows to servers very carefully in some instances (such as when the number of servers is small).

Block-Diagonal Coding



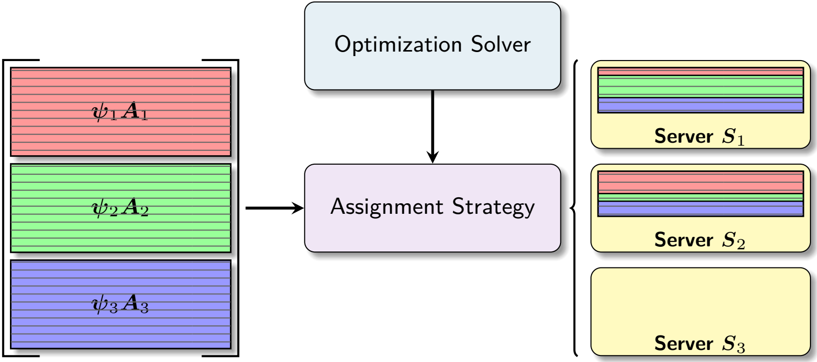
- Block-diagonal encoding with $T = 3$ partitions.
- Need any m/T out of r/T rows from each partition to decode.
- Need to assign coded rows to servers very carefully in some instances (such as when the number of servers is small).
- This assignment can be formulated as an optimization problem.

Block-Diagonal Coding



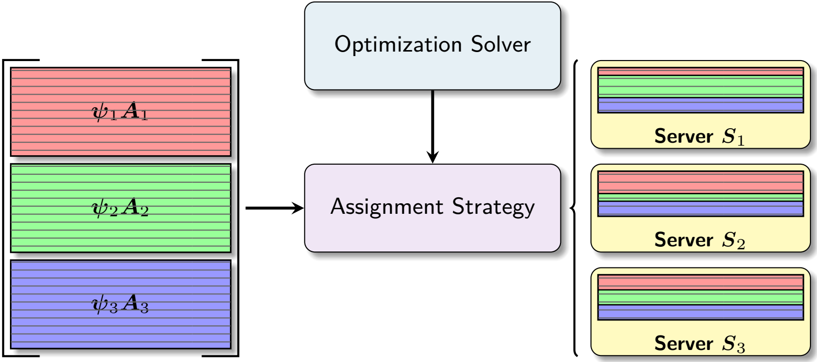
- Block-diagonal encoding with $T = 3$ partitions.
- Need any m/T out of r/T rows from each partition to decode.
- Need to assign coded rows to servers very carefully in some instances (such as when the number of servers is small).
- This assignment can be formulated as an optimization problem.

Block-Diagonal Coding



- Block-diagonal encoding with $T = 3$ partitions.
- Need any m/T out of r/T rows from each partition to decode.
- Need to assign coded rows to servers very carefully in some instances (such as when the number of servers is small).
- This assignment can be formulated as an optimization problem.

Block-Diagonal Coding



- Block-diagonal encoding with $T = 3$ partitions.
- Need any m/T out of r/T rows from each partition to decode.
- Need to assign coded rows to servers very carefully in some instances (such as when the number of servers is small).
- This assignment can be formulated as an optimization problem.

Assignment Matrix

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.
- Store each coded row at 2 servers to enable coded multicasting.

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.
- Store each coded row at 2 servers to enable coded multicasting.

$$\begin{array}{c} (S_1, S_2) \\ (S_1, S_3) \\ (S_1, S_4) \\ (S_2, S_3) \\ (S_2, S_4) \\ (S_3, S_4) \end{array} \begin{pmatrix} \quad \\ \quad \\ \quad \\ \quad \\ \quad \\ \quad \end{pmatrix}$$

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.
- Store each coded row at 2 servers to enable coded multicasting.

$$
\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} (S_1, S_2) \\ (S_1, S_3) \\ (S_1, S_4) \\ (S_2, S_3) \\ (S_2, S_4) \\ (S_3, S_4) \end{array} & \left(\begin{array}{cccccc} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{array} \right)
\end{array}
$$

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.
- Store each coded row at 2 servers to enable coded multicasting.

$$
\begin{array}{cc}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} (S_1, S_2) \\ (S_1, S_3) \\ (S_1, S_4) \\ (S_2, S_3) \\ (S_2, S_4) \\ (S_3, S_4) \end{array} &
\begin{pmatrix}
2 & & & & & \\
 & 2 & & & & \\
 & & 2 & & & \\
 & & & 2 & & \\
 & & & & 2 & \\
 & & & & & 2
\end{pmatrix}
\end{array}
$$

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.
- Store each coded row at 2 servers to enable coded multicasting.
- Assume that S_1 finish first...

$$
\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} (S_1, S_2) \\ (S_1, S_3) \\ (S_1, S_4) \\ (S_2, S_3) \\ (S_2, S_4) \\ (S_3, S_4) \end{array} &
\begin{pmatrix}
2 & & & & & \\
 & 2 & & & & \\
 & & 2 & & & \\
 & & & 2 & & \\
 & & & & 2 & \\
 & & & & & 2
\end{pmatrix}
\end{array}
$$

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.
- Store each coded row at 2 servers to enable coded multicasting.
- Assume that S_1 finish first...

$$\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} (S_1, S_2) \\ (S_1, S_3) \\ (S_1, S_4) \\ (S_2, S_3) \\ (S_2, S_4) \\ (S_3, S_4) \end{array} & \begin{pmatrix} 2 & & & & & \\ & 2 & & & & \\ & & 2 & & & \\ & & & 2 & & \\ & & & & 2 & \\ & & & & & 2 \end{pmatrix}
\end{array}$$

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.
- Store each coded row at 2 servers to enable coded multicasting.
- Assume that S_1 finish first...
- ...and S_2 second.

$$
\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} (S_1, S_2) \\ (S_1, S_3) \\ (S_1, S_4) \\ (S_2, S_3) \\ (S_2, S_4) \\ (S_3, S_4) \end{array} &
\begin{pmatrix}
2 & & & & & \\
 & 2 & & & & \\
 & & 2 & & & \\
 & & & 2 & & \\
 & & & & 2 & \\
 & & & & & 2
\end{pmatrix}
\end{array}
$$

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.
- Store each coded row at 2 servers to enable coded multicasting.
- Assume that S_1 finish first...
- ...and S_2 second.

$$\begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} (S_1, S_2) \\ (S_1, S_3) \\ (S_1, S_4) \\ (S_2, S_3) \\ (S_2, S_4) \\ (S_3, S_4) \end{array} & \left(\begin{array}{cccccc} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{array} \right) \end{array}$$

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.
- Store each coded row at 2 servers to enable coded multicasting.
- Assume that S_1 finish first...
- ...and S_2 second.

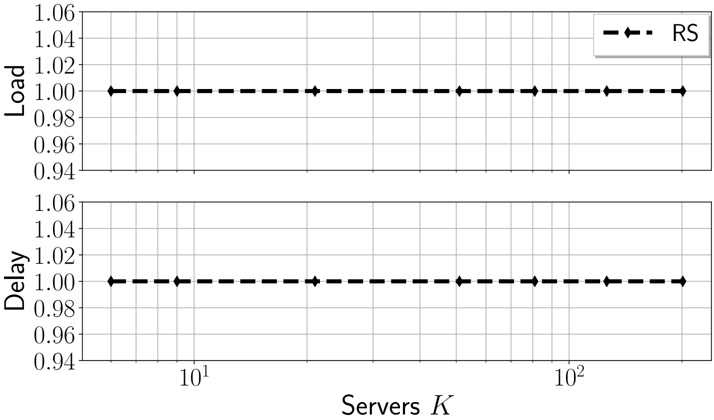
$$
\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} (S_1, S_2) \\ (S_1, S_3) \\ (S_1, S_4) \\ (S_2, S_3) \\ (S_2, S_4) \\ (S_3, S_4) \end{array} &
\begin{pmatrix}
1 & 1 & & & & \\
 & & 1 & 1 & & \\
 & & & & 1 & 1 \\
1 & 1 & & & & \\
 & & 1 & 1 & & \\
 & & & & 1 & 1
\end{pmatrix}
\end{array}
$$

Assignment Matrix

- $K = 4$ servers, $T = 6$ partitions, $m = 6$ source matrix rows, and $r = 12$ coded rows, i.e., code rate $1/2$.
- Store each coded row at 2 servers to enable coded multicasting.
- Assume that S_1 finish first...
- ...and S_2 second.

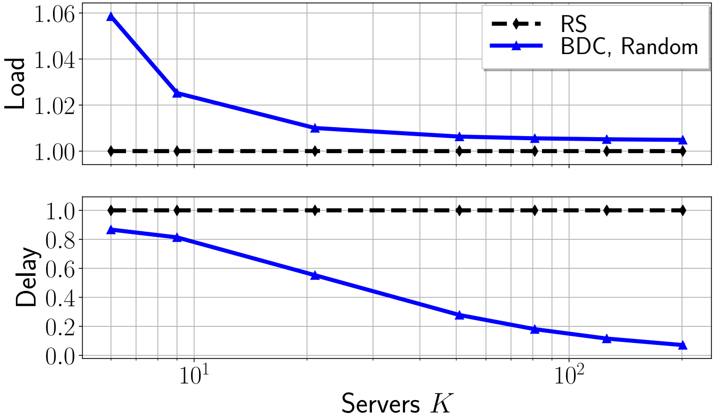
$$
\begin{array}{c c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} (S_1, S_2) \\ (S_1, S_3) \\ (S_1, S_4) \\ (S_2, S_3) \\ (S_2, S_4) \\ (S_3, S_4) \end{array} &
\begin{pmatrix}
1 & 1 & & & & \\
 & & 1 & 1 & & \\
 & & & & 1 & 1 \\
1 & 1 & & & & \\
 & & 1 & 1 & & \\
 & & & & 1 & 1
\end{pmatrix}
\end{array}
$$

Numerical results



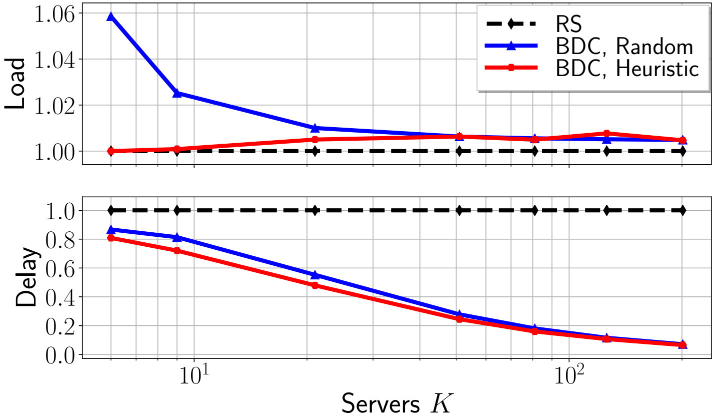
- 2000 rows assigned to each server, $n = 10000$ columns, $m/T = 10$ rows per partition, $N = 2K/3$ vectors, and code rate $m/r = 2/3$.

Numerical results



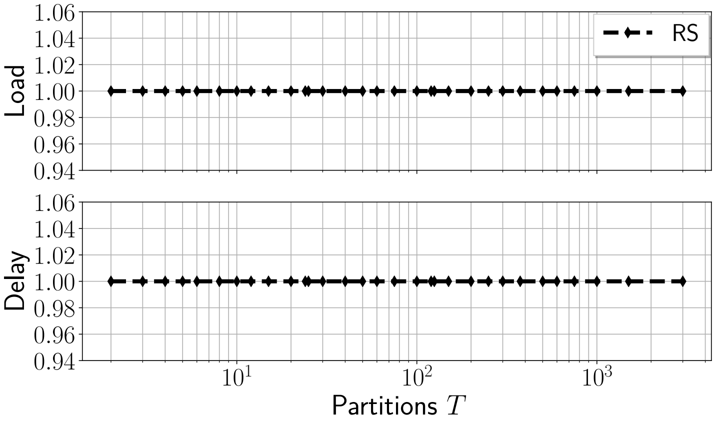
- 2000 rows assigned to each server, $n = 10000$ columns, $m/T = 10$ rows per partition, $N = 2K/3$ vectors, and code rate $m/r = 2/3$.

Numerical results



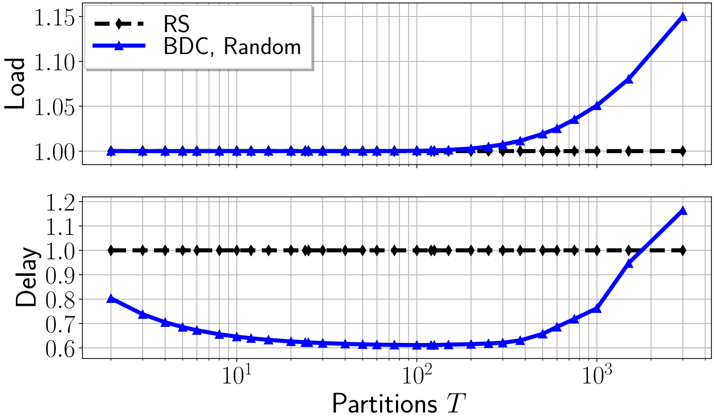
- 2000 rows assigned to each server, $n = 10000$ columns, $m/T = 10$ rows per partition, $N = 2K/3$ vectors, and code rate $m/r = 2/3$.

Numerical results



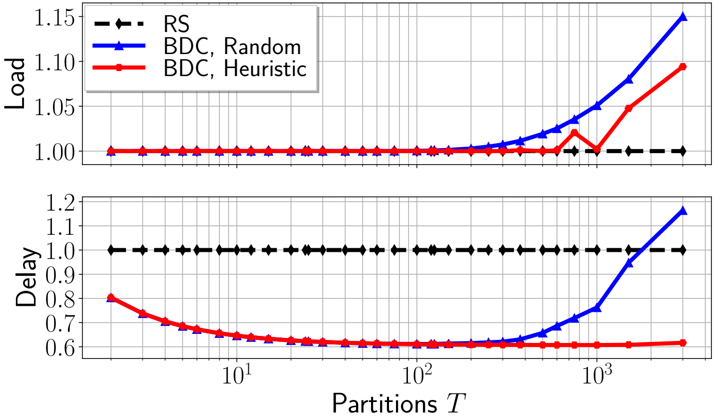
- $m = 6000$ source matrix rows, $n = 6000$ columns, $N = 6$ vectors, $K = 9$ servers, and code rate $m/r = 2/3$.

Numerical results



- $m = 6000$ source matrix rows, $n = 6000$ columns, $N = 6$ vectors, $K = 9$ servers, and code rate $m/r = 2/3$.

Numerical results



- $m = 6000$ source matrix rows, $n = 6000$ columns, $N = 6$ vectors, $K = 9$ servers, and code rate $m/r = 2/3$.

Optimal Assignment

Theorem

- Up to a given number of partitions T, the
  - communication load, and the
  - computational delay of the matrix multiplication, i.e., not taking the decoding delay into account,
- of the BDC scheme are equal to those of the scheme by Li *et al.*.

Optimal Assignment

Theorem

- Up to a given number of partitions T, the
  - communication load, and the
  - computational delay of the matrix multiplication, i.e., not taking the decoding delay into account,
- of the BDC scheme are equal to those of the scheme by Li *et al.*.

The overall computational delay of our scheme is much lower than that of the scheme by Li *et al.* due to its lower decoding complexity.

Conclusion

Take-home message...

- The delay of the scheme by Li *et al.* is dominated by the decoding delay. Our scheme addresses this issue with little to no impact on the load.

Conclusion

Take-home message...

- The delay of the scheme by Li *et al.* is dominated by the decoding delay. Our scheme addresses this issue with little to no impact on the load.
  - $\approx 40\%$ reduced delay over the scheme by Li *et al.* for a matrix with 6000 rows and columns with no impact on load.

Conclusion

Take-home message...

- The delay of the scheme by Li *et al.* is dominated by the decoding delay. Our scheme addresses this issue with little to no impact on the load.
  - $\approx 40\%$ reduced delay over the scheme by Li *et al.* for a matrix with 6000 rows and columns with no impact on load.
  - $> 90\%$ reduced delay over the scheme by Li *et al.* for a matrix with 134000 rows and 10000 columns with $< 1\%$ increased load.

Conclusion

Take-home message...

- The delay of the scheme by Li *et al.* is dominated by the decoding delay. Our scheme addresses this issue with little to no impact on the load.
  - $\approx 40\%$ reduced delay over the scheme by Li *et al.* for a matrix with 6000 rows and columns with no impact on load.
  - $> 90\%$ reduced delay over the scheme by Li *et al.* for a matrix with 134000 rows and 10000 columns with $< 1\%$ increased load.

- Full version of the paper will soon be available on arXiv.

Conclusion

Take-home message...

- The delay of the scheme by Li *et al.* is dominated by the decoding delay. Our scheme addresses this issue with little to no impact on the load.
  - $\approx 40\%$ reduced delay over the scheme by Li *et al.* for a matrix with 6000 rows and columns with no impact on load.
  - $> 90\%$ reduced delay over the scheme by Li *et al.* for a matrix with 134000 rows and 10000 columns with $< 1\%$ increased load.

- Full version of the paper will soon be available on arXiv.
- Encoding delay, LT codes under inactivation decoding...

Conclusion

Take-home message...

- The delay of the scheme by Li *et al.* is dominated by the decoding delay. Our scheme addresses this issue with little to no impact on the load.
  - $\approx 40\%$ reduced delay over the scheme by Li *et al.* for a matrix with 6000 rows and columns with no impact on load.
  - $> 90\%$ reduced delay over the scheme by Li *et al.* for a matrix with 134000 rows and 10000 columns with $< 1\%$ increased load.

- Full version of the paper will soon be available on arXiv.
- Encoding delay, LT codes under inactivation decoding...
- LT codes may reduce delay further at the expense of increased load.

Conclusion

Take-home message...

- The delay of the scheme by Li *et al.* is dominated by the decoding delay. Our scheme addresses this issue with little to no impact on the load.
  - $\approx 40\%$ reduced delay over the scheme by Li *et al.* for a matrix with 6000 rows and columns with no impact on load.
  - $> 90\%$ reduced delay over the scheme by Li *et al.* for a matrix with 134000 rows and 10000 columns with $< 1\%$ increased load.

- Full version of the paper will soon be available on arXiv.
- Encoding delay, LT codes under inactivation decoding...
- LT codes may reduce delay further at the expense of increased load.
- Slides and code on Github: github.com/severinson/coded-computing-tools

References

[1] K. Lee et al. "Speeding up distributed machine learning using codes". In: *Proc. IEEE Int. Symp. Inf. Theory*. Barcelona, Spain, July 2016, pp. 1143–1147. DOI: 10.1109/ISIT.2016.7541478.

[2] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr. "Coded MapReduce". In: *Proc. Annual Allerton Conf. Commun., Control, and Computing*. Monticello, IL, Sept. 2015, pp. 964–971. DOI: 10.1109/ALLERTON.2015.7447112.

[3] Songze Li, Mohammad Ali Maddah-Ali, and Amir Salman Avestimehr. "A Unified Coding Framework for Distributed Computing with Straggling Servers". In: *Proc. Workshop Network Coding and Appl.* Washington, DC, Dec. 2016. DOI: 10.1109/GLOCOMW.2016.7848828.