Coding for Distributed Computing

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

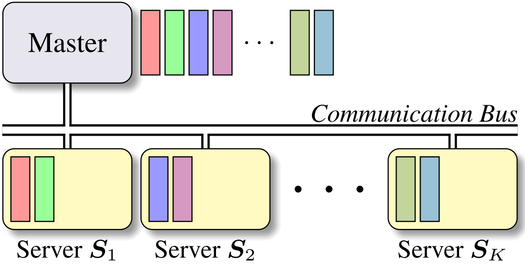
† Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden
‡ University of Bergen/Simula Research Lab, Bergen, Norway

Finse, May 09, 2018

simula@uib

CHALMERS

Motivation



Challenges

- Straggler problem: May induce a large computational delay.
- Bandwidth scarcity: Need to reduce the communication load.

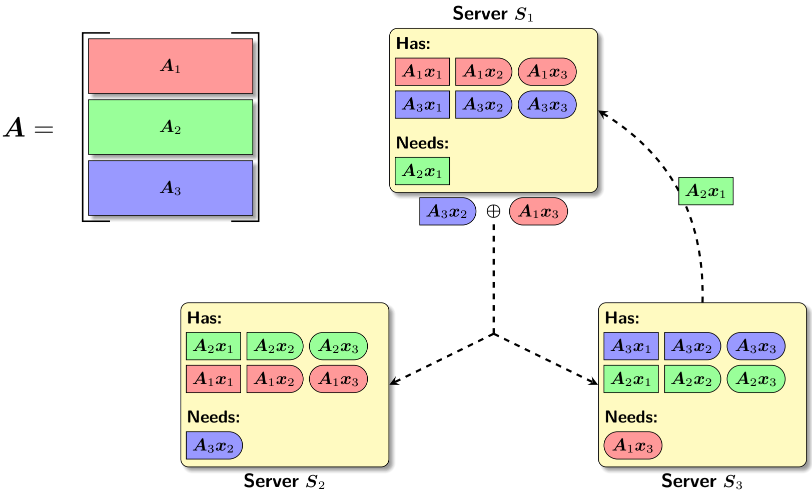
Problem addressed: Matrix multiplication

- 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.

Bandwidth Scarcity

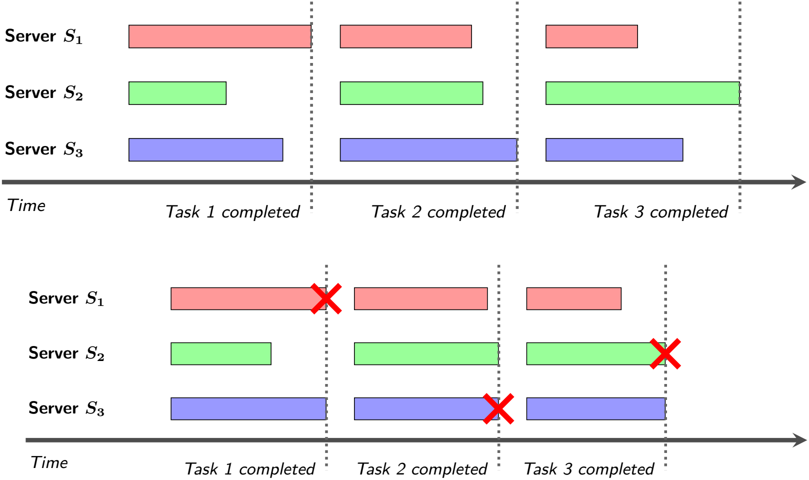
(Coded MapReduce, Li *et al.*, 2015)

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



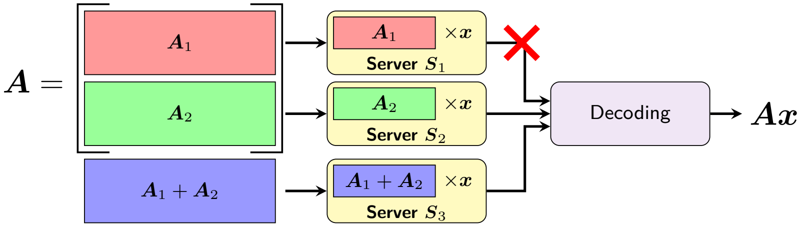
The straggler problem

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



The Straggler Problem

$$y = Ax$$

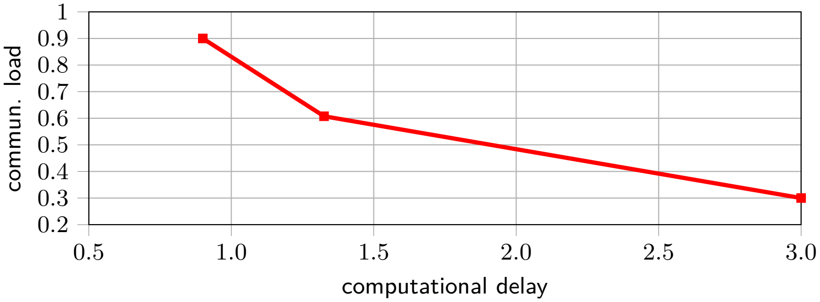


In general

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

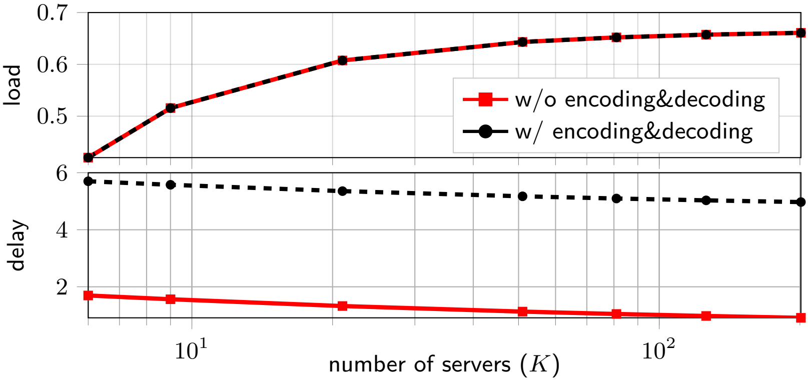
Coding for distributed computing

- [Lee *et al.* '17]: Introduce redundant computations using MDS codes to alleviate the straggler problem.
- [Li, Maddah-Ali, Avestimehr '17]: A fundamental tradeoff between computational delay and communication load. A unified coding framework trading higher computational delay for lower communication load.



Unified coding framework [Li, Maddah-Ali, Avestimehr '17]

- Encode the columns of $\boldsymbol{A} \in \mathbb{F}^{m \times n}$ using an (r, m) MDS code by multiplying $\boldsymbol{A}$ by an $r \times n$ encoding matrix $\boldsymbol{\Psi}_{\mathsf{MDS}}$, i.e., $\boldsymbol{C} = \boldsymbol{\Psi}_{\mathsf{MDS}}\boldsymbol{A}$.
- Code length r proportional to number of rows of $\boldsymbol{A}$ → high overall delay!



- $\boldsymbol{A}$ with $n = 10000$ columns and $m = 2000K/3$ rows, $N = 2000K/3$ vectors, and code rate $2/3$ (2000 rows assigned to each server).

In this talk

Two coding schemes to reduce the overall computational delay

- Block-diagonal coding scheme, based on a block-diagonal encoding matrix and shorter MDS codes.
- LT code-based scheme under inactivation decoding.

Outcome

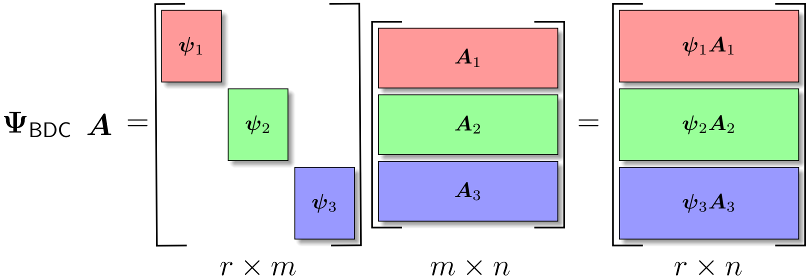
- Block-diagonal coding scheme: Significantly lower overall computational delay than the scheme by [Li, Maddah-Ali, Avestimehr '17] with no or little impact on communication load.
- LT code-based scheme: Very good performance when requiring to meet a deadline with high probability, at the expense of an increased communication load.

Block-diagonal coding scheme

Idea

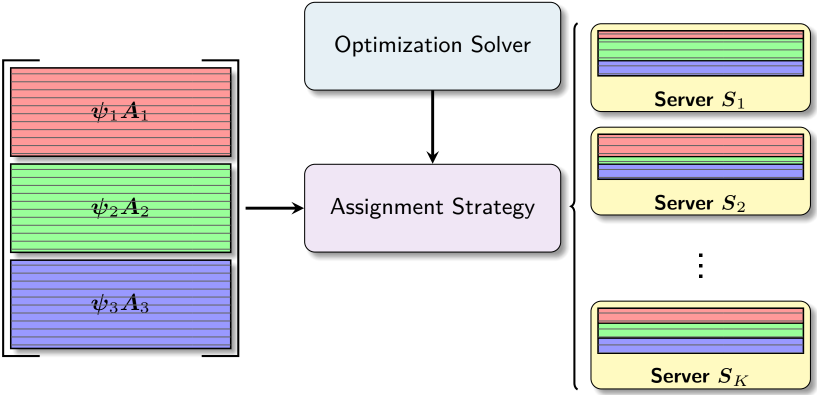
- Partition $\boldsymbol{A}$ into T disjoint submatrices and apply smaller MDS codes to each submatrix,

$$\boldsymbol{C} = \boldsymbol{\Psi}_{\mathsf{BDC}}\boldsymbol{A}, \quad \boldsymbol{\Psi}_{\mathsf{BDC}} = \begin{bmatrix} \boldsymbol{\psi}_1 & & \\ & \ddots & \\ & & \boldsymbol{\psi}_T \end{bmatrix}, \quad \boldsymbol{\psi}_i : \left(\frac{r}{T}, \frac{m}{T}\right) \text{ MDS code.}$$



- Need any m/T out of r/T rows from each partition to decode.

Assignment of coded rows to servers



- 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.

Lossless partitioning

Theorem

For $T \leq r/\binom{K}{\mu q}$, there exists an assignment matrix such that the communication load and the computational delay (not taking encoding/decoding delay into account) are equal to those of the unpartitioned scheme by [Li, Maddah-Ali, Avestimehr '17].

However...

The overall computational delay of the block-diagonal coding scheme is much lower than that of the scheme by Li *et al.* due to its lower encoding and decoding complexity.

Luby-transform code-based scheme

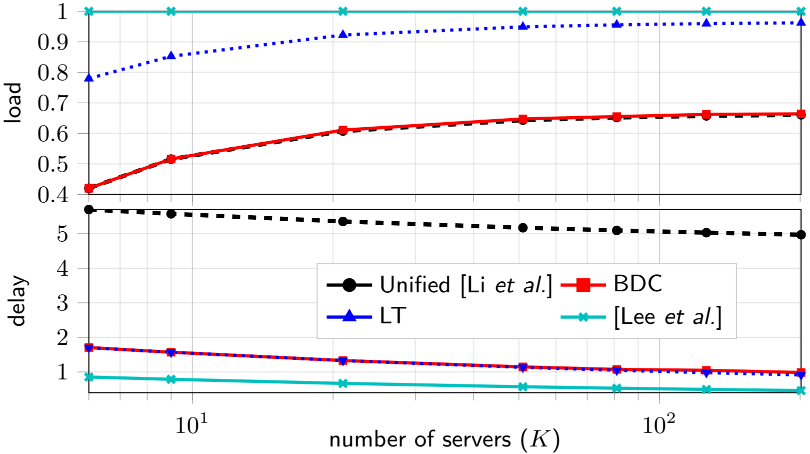
LT code-based scheme

- Encode $\boldsymbol{A}$ as $\boldsymbol{C} = \boldsymbol{\Psi}_{\mathsf{LT}}\boldsymbol{A}$; $\boldsymbol{\Psi}_{\mathsf{LT}}$ corresponds to an LT code of fixed rate.
- Decode the LT code using inactivation decoding.

Code design

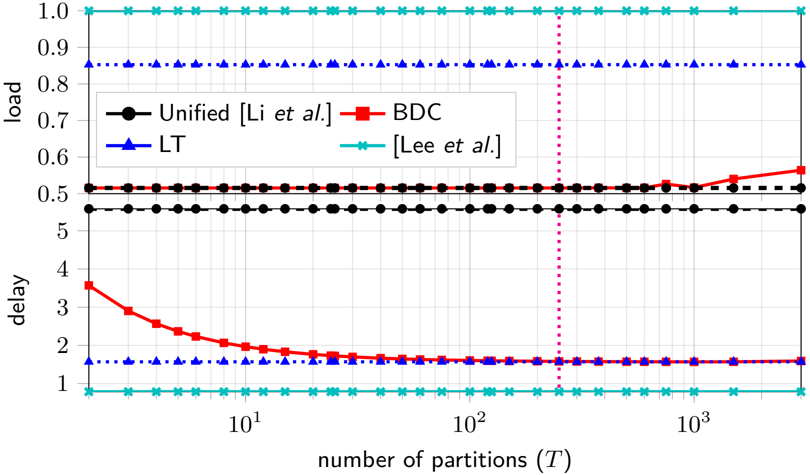
- Design the LT code for a minimum overhead ϵ_{min} and a target failure probability $P_{\mathsf{f,target}}$, such that $P_{\mathsf{f}}(\epsilon_{\mathsf{min}}) \leq P_{\mathsf{f,target}}$.
- Increasing ϵ_{min} leads to lower encoding/decoding complexity but increased communication load and may require waiting for more servers $\longrightarrow$ optimal ϵ_{min} depends on the scenario.
- For a given ϵ_{min} and $P_{\mathsf{f,target}}$, optimize the LT code so that the decoding complexity is minimized: for a fixed computational delay of $\boldsymbol{C}\boldsymbol{x}_1, \ldots, \boldsymbol{C}\boldsymbol{x}_N$, minimize the computational delay of the decoding phase.

Computational delay and communication load



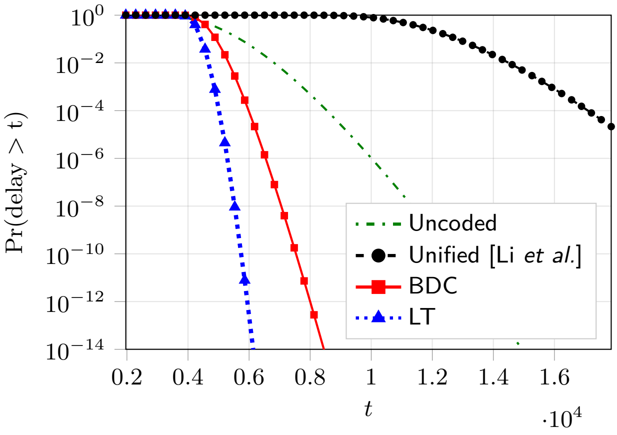
- $\boldsymbol{A}$ with $n = 10000$ columns and $m = 2000K/3$ rows. $N = 2000K/3$ vectors. Rate $2/3$, i.e., 2000 rows assigned to each server and $m/T = 10$ rows per partition.

Performance as a function of the number of partitions



- $\boldsymbol{A}$ with $m = 6000$ rows and $n = 6000$ columns, $N = 6$ vectors, $K = 9$ servers, and code rate $2/3$.

Distributed computing under a deadline



- $\boldsymbol{A}$ with $m = 134000$ rows and $n = 10000$ columns, $N = 134000$ vectors, $K = 201$ servers, $T = 13400$ partitions, and code rate $2/3$.

Conclusion

Take-home message...

- The encoding and decoding delay may contribute significantly to the overall computational delay.
- The BDC scheme yields significantly lower computational delay (up to 70%–80%) with no or little impact on the communication load.
- The LT code-based scheme achieves very good performance when needing to meet a deadline with high probability.

- Paper available in arxiv:
  A. Severinson, A. Graell i Amat, and E. Rosnes, "Block-Diagonal and LT Codes for Distributed Computing With Straggling Servers".
- Code on Github: github.com/severinson/coded-computing-tools

One More Thing...