Coded Computing for Straggler Mitigation, Security and Privacy

EE605 Error Correcting Codes

Param Rathour 190070049
Anupam Nayak 19D070010

Department of Electrical Engineering
Indian Institue of Technology Bombay

Autumn 2021-22

Outline

1. Straggler Mitigation in Distributed Matrix Multiplications
   - Problem Formulation
   - Computation Strategy
2. Lagrange Coded Computing – Going beyond Matrix Algebra
   - Polynomial Evaluations
   - Secure and Private Multiparty Computing

Distributed Matrix Multiplication

Problem formulation

We have two input matrices

- $A \in \mathbb{F}_q^{s \times r}$
- $B \in \mathbb{F}_q^{s \times t}$ for some sufficiently large finite field $\mathbb{F}_q$
- **To compute** $C = A^T B$. It is implicitly assumed in general that one of these matrices is tall.

Each worker has to be assigned fraction of the coded a fraction of the submatrix

- Each of the N workers stores $\frac{1}{m}$ fraction of A and $\frac{1}{n}$ fraction of B where $m, n \in \mathbb{N}$
- Thus $A_i \in \mathbb{F}_q^{s \times \frac{r}{m}}$ and $B_i \in \mathbb{F}_q^{s \times \frac{t}{n}}$

Idea is for the master to use something like an MDS encoding to generate each of these sub matrices such that it has to wait only for k of these workers(fastest) to generate the output in order to uniquely identify the product.

Problem Formulation

Key Ideas

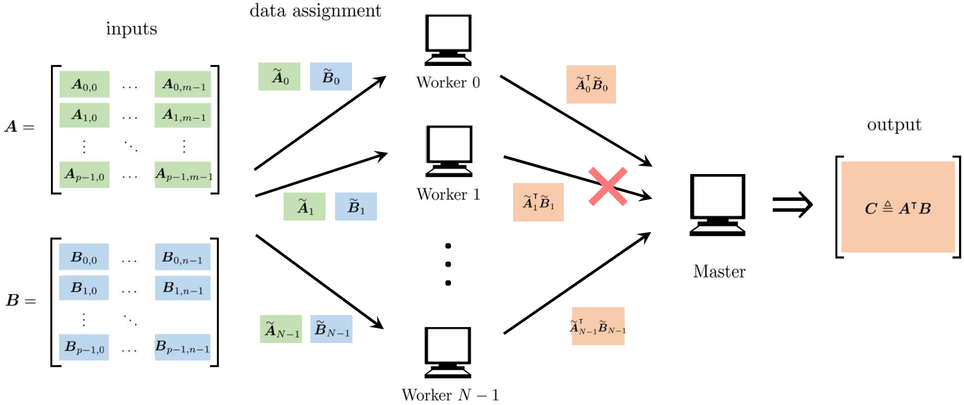


Figure: Overview of the distributed matrix multiplication problem[1].

[1] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," 2020.

Computation Strategy

Choice of functions

A computation strategy is defined as a set of $2N$ functions

$$f = (f_0, f_1 .... f_{N-1})$$

$$g = (g_0, g_1 .... g_{N-1})$$

that are used to compute $A_i = f_i(A)$, $B_i = g_i(B)$ $\forall i \in \{0, 1, 2, ...N-1\}$

For any integer k we say that the system is k recoverable if the master can recover the product C using output from any k workers

We define $k(f,g)$ as the least integer k for which the system defined by f,g is k recoverable

Computation Strategy

Optimum Recovery Threshold

The lowest among the recovery thresholds across all computation strategies

$$K^* = \min_{f,g} k(f,g)$$

State-of-the-art schemes include

- 1D MDS scheme

$$K_{1D\ MDS} = N - \frac{N}{n} + m$$

- In the same paper an alternative scheme for the special case of $m = n$ referred to as the product code achieves a threshold of

$$K_{prod} = 2(m-1)\sqrt{N} - (m-1)^2 + 1$$

Main Result

Theorem

The distributed matrix multiplication problem of computing $A^T B$ described above has a minimum recovery threshold of

$$K^* = mn$$

Further $\exists$ a computation strategy referred to as polynomial code which achieves the above K^* which allows for efficient decoding at the master node with computational complexity of polynomial decoding with mn points.

Decoding polynomials codes essentially a polynomial interpolation problem, which can be solved in time almost linear to the input size.This is enabled by designing the computing strategies such that the computed products form a Reed-Solomon code.

Comparison of the four thresholds

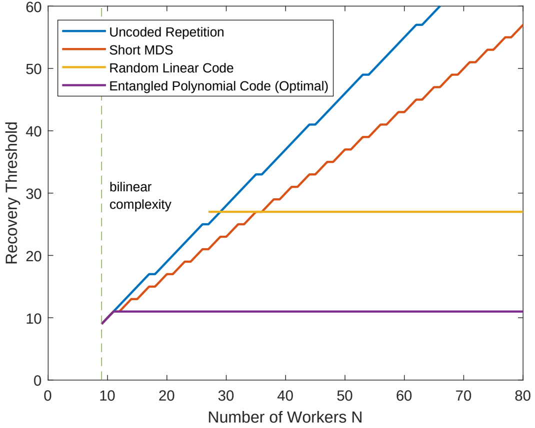


Figure: Comparison[2]

[2] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," 2020.

Proof I

A sketch of the proof

Given parameters $\alpha, \beta \in \mathbb{N}$, we define the (α, β)- polynomial code $\forall i \in \{0, 1....N-1\}$ as

$$\overline{A_i} = \sum_{j=0}^{m-1} A_j x_i^{j\alpha}$$

$$\overline{B_i} = \sum_{j=0}^{n-1} B_j x_i^{j\beta}$$

$$\overline{C_i} = \overline{A_i}^T \overline{B_i} = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} A_j^T B_k x_i^{j\alpha + k\beta}$$

Note here we need to carefully choose α and β such that no two terms have the same power of x. One such choice being $\alpha = 1$, $\beta = m$. we define $h(x)$ as follows

Proof II

$$h(x) = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} A_j^T B_k x^{j+k\beta}$$

This is polynomial of degree $mn - 1$. Since x_i are chosen to be different in order to recover C we need the output from any mn workers which is essentially interpolating using mn points, For even less complex decoding we may use the Reed Solomon decoding algorithm.

So far we have a scheme that achieves the bound in the theorem. We need to prove that we require output from atleast mn workers to recover C in order to prove optimality

Proof III

A sketch of this proof is as follows

- WLOG Let A be an arbitrary fixed tall matrix ($s \geq r$) and B is sampled from a uniform distribution over $\mathbb{F}_q^{s \times t}$
- Thus one can check the distribution of $C = A^T B$ will be uniform over $\mathbb{F}_q^{r \times t}$
- This means we need to recover a random variable with entropy $H(C) = rt \log q$
- Since each worker outputs $\frac{rt}{mn}$ elements of $\mathbb{F}_q$ it provides atmost $\frac{rt}{mn} \log q$ bits of information

  hence $K > mn$

Performance of the polynomial code on other evaluation metrics

- **Computation latency** is defined as the amount of time required for the master to collect enough information to decode C, For any other computation strategy we have

$$T \geq T_{poly}$$

- **Probability of failure given a deadline** is defined as the probability that the master does not receive enough information to decode C at a predefined time t

$$P(T > t) \geq P(T_{poly} > t)$$

- **Communication load** is defined as the minimum number of bits needed to be extracted in order to complete the computation. The below mentioned bound is achieved by the polynomial code.

$$L^* = rt \log_2 q$$

Polynomial Evaluation

Problem Formulation

We have a distributed computing environment with a master and N workers

- **Dataset** $X = (X_1, \ldots, X_k)$ where X_i is a element of a vector space $\mathbb{V}$ over $\mathbb{F}$
- $f : \mathbb{V} \to \mathbb{U}$ is a multivariate polynomial with vector coefficients and degree $= \deg f$
- **To compute** $Y_1 \triangleq f(X_1), \ldots, Y_K \triangleq f(X_K)$

Each worker has already stored a fraction of the coded dataset prior to computation

- The i^{th} worker stores $\tilde{X}_i \triangleq g_i(X_1, \ldots, X_K)$, where g_i is a (possibly random) function, refered to as the encoding function of that worker. ($i \in [N]$ and $[N] \triangleq \{1, \ldots, N\}$)
- Each worker $i \in [N]$ computes $\tilde{Y}_i \triangleq f(\tilde{X}_i)$ and returns the result to the master.

The master waits for a subset of fastest workers and then decodes $Y_1, \ldots, Y_K$.
Linear encoding strategy gives simple yet concrete implmentation,

Some Polynomial Evaluations tasks

Example (Linear Computation)

• Compute $A\vec{b}$ for some dataset $A = \{A_i\}_{i=1}^K$ and vector $\vec{b}$
Let $\mathbb{V}$ be the space of matrices over $\mathbb{F}$, $\mathbb{U}$ be the space of vectors over $\mathbb{F}$, X_i be A_i, and $f(X_i) = X_i \cdot \vec{b}$ for all $i \in [K]$. (suitable dimensions to be assigned to $\mathbb{V}, \mathbb{U}$)

Example (Bilinear Computation)

• Compute element-wise products $\{A_i \cdot B_i\}_{i=1}^K$ of two matrices $\{A_i\}_{i=1}^K$ and $\{B_i\}_{i=1}^K$.
Let $\mathbb{V}$ be the space of pairs of two matrices, $\mathbb{U}$ be the space of matrices, $X_i = (A_i, B_i)$, and $f(X_i) = A_i \cdot B_i$ for all $i \in [K]$. (suitable dimensions to be assigned to $\mathbb{V}, \mathbb{U}$)

Result on Minimum Workers required for recovery

Theorem

Given a number of workers N and a dataset $X = (X_1, \dots, X_K)$, for distributedly computing f, the minimum recovery threshold is given by

$$K^* = \begin{cases} (K-1)\deg f + 1 & K \deg f - 1 \leq N \\ N - \lfloor N/K \rfloor + 1 & \textit{else} \end{cases} \tag{1}$$

Theorem

The Lagrange Coded Computing is optimal i.e. it minimizes the recovery threshold

Proof.

Coming up. □

Polynomial Interpolation

Given a set of $N+1$ data points (x_i, y_i) where no two x_i are the same, a polynomial $p : \mathbb{R} \to \mathbb{R}$ is said to interpolate the data if $p(x_j) = y_j$ for each $j \in [N+1]$

Construction using System of Linear Equations

Suppose that the interpolation polynomial is in the form

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

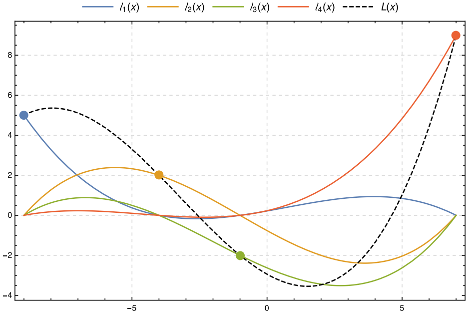
Now, we can frame a system of linear equations as $p(x_i) = y_i \quad$ for $i \in [N+1]$

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \dots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \dots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Lagrange Interpolation[3]

$$p(x) = \frac{(x - x_1)(x - x_2)\cdots(x - x_n)}{(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_n)} y_0 + \cdots + \frac{(x - x_0)(x - x_1)\cdots(x - x_{n-1})}{(x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1})} y_n$$

$$p(x) = \sum_{i=0}^{n} \left(\prod_{\substack{0 \le j \le n \\ j \ne i}} \frac{x - x_j}{x_i - x_j} \right) y_i \qquad (2)$$



[3] https://commons.wikimedia.org/wiki/File:Lagrange_polynomial.svg

Lagrange Coded Computing

Key Ideas

- The Lagrange interpolation polynomial is used to create encoding of the input dataset inserting computational redundancy in a coded form across the workers.
- The computations at the each worker amount to evaluations of a composition of this polynomial with the desired function f resulting in another polynomial h_i.
- Decode $Y_1, \ldots, Y_K$ using only K^* of h_i's by evaluating each at certain points.

Lagrange Coded Computing

Encoding

- Select any K distinct elements $\beta_1, \ldots, \beta_K$ from $\mathbb{F}$, and find a polynomial $u : \mathbb{F} \to \mathbb{V}$ of degree $K-1$ such that $u(\beta_i) = X_i$ for $i \in [K]$.
  This can be accomplished by Lagrange interpolation polynomial

$$u(z) \triangleq \sum_{j \in [K]} X_j \cdot \prod_{k \in [K] \setminus \{j\}} \frac{z - \beta_k}{\beta_j - \beta_k} \tag{3}$$

- Now, select N distinct elements $\alpha_1, \ldots, \alpha_N$ from $\mathbb{F}$ and encode the input variables by letting $\tilde{X} = u(\alpha_i)$ for $i \in [N]$

$$\tilde{X}_i = g_i(X) = u(\alpha_i) \triangleq \sum_{j \in [K]} X_j \cdot \prod_{k \in [K] \setminus \{j\}} \frac{\alpha_i - \beta_k}{\beta_j - \beta_k} \tag{4}$$

Lagrange Coded Computing

Decoding

- Each worker i computes $\tilde{Y}_i = f(\tilde{X}_i) = f(u(\alpha_i))$ and sends $\tilde{Y}_i$ to the master.
- This composition $f(u(z))$ is also a polynomial with degree $\leq (K-1)\deg f$.
- Now, any $(K-1)\deg f+1$ workers return the evaluations at $(K-1)\deg f+1$ points. This gives a unique $f(u(z))$ which can be interpolated using Lagrange polynomials.
- Then, the master evaluates it at β_i for every $i \in [K]$ to obtain $f(u(\beta_i)) = f(X_i)$,

Note that if, number of workers are small $(N < K\deg f - 1)$, K^* can be easily achieved by replicating every X_i by atleast $\lfloor N/K \rfloor$ times. Now, every set of $N - \lfloor N/K \rfloor + 1$ computation contains at least one copy of $f(X_i)$ for every i.

Secure and Private Multiparty Computing

Overview

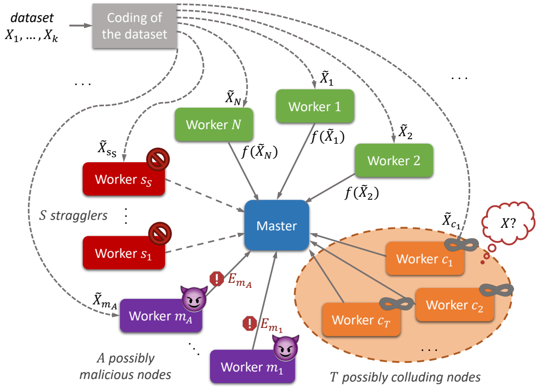


Figure: Overview of Secure and Private Multiparty Computing[4]

[4] J. S. Qian Yu, Netanel Raviv and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," 2018

Secure and Private Multiparty Computing

Terminology

- **Resiliency** (robustness against stragglers) In a S-*resilient* system, the master must be able to obtain the correct values of $Y_1, \ldots, Y_K$ even if up to S workers delay/fail.
- **Security** (robustness against adversaries) In a A-*secure* system, the master must be able to obtain correct values of $Y_1, \ldots, Y_K$ even if up to A workers return arbitrarily erroneous results.
- **Privacy** (robustness against collusion) In a T-*private* system, the workers cannot infer anything about the content of the dataset, even if up to T of them collude, Formally, for every $\mathcal{T} \subseteq [N]$ of size at most T, we must have $I(X; \tilde{X}_{\mathcal{T}}) = 0$

The tuple (S, A, T) is achievable if there exists an encoding and decoding scheme that can complete the computations in the presence of up to S stragglers, up to A adversarial workers, whilst keeping the dataset private against sets of up to T colluding workers.

Main Result

The below theorem characterizes the region for (S, A, T) that LCC achieves

Theorem

Given a number of workers N and a dataset $X = (X_1, \ldots, X_K)$, LCC provides an S-resilient, A-secure, and T-private scheme for computing $\{f(X_i)\}_{i=1}^{K}$ for any polynomial f, as long as

$$(K + T - 1) \deg f + S + 2A + 1 \leq N. \tag{5}$$

Interesting Fact

One additional worker can increase its resiliency to stragglers by 1, or increase its robustness to adversaries by 1/2, while maintaining the privacy constraint. Sounds familiar?

Lagrange Coded Computing

Encoding

- Select any $K+T$ distinct elements $\beta_1, \ldots, \beta_{K+T}$ from $\mathbb{F}$, and find a polynomial $u : \mathbb{F} \to \mathbb{V}$ of degree $K+T-1$ such that $u(\beta_i) = X_i$ for $i \in [K]$ and $u(\beta_i) = Z_i$ for $i \in \{K+1, \ldots, K+T\}$ where are Z_i's are chosen randomly from $\mathbb{V}$.

$$u(z) \triangleq \sum_{j\in[K]} X_j \cdot \prod_{k\in[K+T]\setminus\{j\}} \frac{z-\beta_k}{\beta_j-\beta_k} + \sum_{j=K+1}^{K+T} Z_j \cdot \prod_{k\in[K+T]\setminus\{j\}} \frac{z-\beta_k}{\beta_j-\beta_k} \quad (6)$$

- Now, select N distinct elements $\alpha_1, \ldots, \alpha_N$ from $\mathbb{F}$ such that $\{\alpha_i\}_{i=1}^N \cap \{\beta_j\}_{j=1}^N = \varnothing$ and encode the input variables by letting $\tilde{X} = u(\alpha_i)$ for $i \in [N]$

$$\tilde{X}_i = g_i(X) = u(\alpha_i) = (X_1, \ldots, X_K, Z_{K+1}, \ldots, Z_{K+T}) \cdot U_i \quad (U \in \mathbb{F}_q^{(K+T)\times N}) \quad (7)$$

Where $U_{ij} \triangleq \prod_{l\in[K+T]\setminus\{i\}} \frac{\alpha_j-\beta_l}{\beta_i-\beta_l}$

Lagrange Coded Computing

Decoding

- Each worker i computes $\tilde{Y}_i = f(\tilde{X}_i) = f(u(\alpha_i)$ and sends $\tilde{Y}_i$ to the master.
- This composition $f(u(z))$ is also a polynomial with degree $\leq (K + T - 1) \deg f$.
- The master obtains $N - S$ evaluations of $f(u(z))$, at most A of which are incorrect.
- The master can obtain all coefficients of $f(u(z))$ by applying Reed-Solomon decoding as $N \geq (K + T - 1) \deg(f) + S + 2A + 1$
- Then, the master evaluates it at β_i for every $i \in [K]$ to obtain $f(u(\beta_i)) = f(X_i)$,

Hence, we have shown that the above scheme is S-resilient and A-secure.

Recent Works and Open Problems

- Developing efficient and straggler resilient “master-less” systems.
  - Current state of the art assumes availability of master
  - All nodes are identical, and no single node may be able to store all the data or perform all the encoding/decoding.
- Beyond polynomial computations.
  - Going beyond polynomial computations is a very important and challenging research
  - Impacts application domains (eg. machine learning with non-linear threshold functions)
- Application to blockchain systems
  - Today’s blockchain designs suffer from a trilemma claiming that no blockchain system can simultaneously achieve decentralization, security, and performance scalability.
  - Coded computing can provide an effective approach for overcoming such barriers.

For Further Reading

- Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.
- J. S. Qian Yu, Netanel Raviv and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *CoRR*, vol. abs/1806.00939, 2018.
- K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 2418–2422, 2017.
- S. Li and S. Avestimehr, "Coded computing: Mitigating fundamental bottlenecks in large-scale distributed computing and machine learning," *Foundations and Trends® in Communications and Information Theory*, vol. 17, no. 1, pp. 1–148, 2020.