Distributed Matrix Multiplication

Umberto Martínez-Peñas
University of Valladolid (UVa)

SecureCAT Workshop,

Aguilar de Campoo 2023

Distributed Matrix Multiplication

- Let $A \in \mathbb{F}^{s \times r}$ and $B \in \mathbb{F}^{s \times t}$, for $r, s, t \in \mathbb{Z}_+$ and a field $\mathbb{F}$.
- We want to compute $C = A^\intercal B$ in a distributed way.
- If we have *mn* workers, we divide

$$A = (A_0, A_1, \ldots, A_{m-1}) \quad \text{and} \quad B = (B_0, B_1, \ldots, B_{n-1}),$$

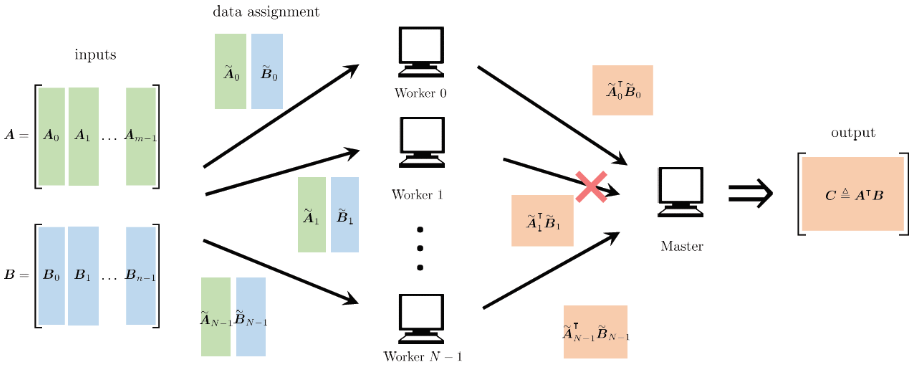
  with appropriate sizes $A_i \in \mathbb{F}^{s \times r'}$ and $B_i \in \mathbb{F}^{s \times t'}$, $r = mr'$ and $t = nt'$.
- Each worker computes a smaller product $A_i^\intercal B_j$, and we recover C by appending these products, since

$$C = A^\intercal B = \begin{pmatrix} A_0^\intercal B_0 & A_0^\intercal B_1 & \ldots & A_0^\intercal B_{n-1} \\ A_1^\intercal B_0 & A_1^\intercal B_1 & \ldots & A_1^\intercal B_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m-1}^\intercal B_0 & A_{m-1}^\intercal B_1 & \ldots & A_{m-1}^\intercal B_{n-1} \end{pmatrix}.$$

Distributed Matrix Multiplication

- In this way, we parallelize the multiplication of two large matrices.
- A typical problem is that some workers may take too long to perform the computation (stragglers).
- In fact, the stragglers may take orders of magnitude longer, and thus they are considered non-responsive.
- However, in the previous parallelization method, the output of every worker is necessary to recover the whole product $C = A^{\intercal} B$.
- Solution: Error-correcting codes.

Distributed Matrix Multiplication



Polynomial Codes

- Polynomial codes are essentially Reed–Solomon codes, but with the previous matrix subdivision and appropriate degree choices.
- We have N workers and divide

$$A = (A_0, A_1, \ldots, A_{m-1}) \quad \text{and} \quad B = (B_0, B_1, \ldots, B_{n-1}),$$

  and as before, we only need to compute $A_i^\intercal B_j$ for all i, j.
- For $\alpha, \beta \in \mathbb{Z}_+$, we define the (α, β)-polynomial code by

$$\widetilde{A}_i = \sum_{j=0}^{m-1} A_j x_i^{\alpha j} \quad \text{and} \quad \widetilde{B}_i = \sum_{k=0}^{n-1} B_k x_i^{\beta k},$$

  for distinct points $x_0, x_1, \ldots, x_{N-1} \in \mathbb{F}$.
- Now, the ith worker computes

$$\widetilde{C}_i = \widetilde{A}_i^\intercal \widetilde{B}_i = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} A_j^\intercal B_k x_i^{\alpha j + \beta k}.$$

Polynomial Codes

- Now, the ith worker computes

$$\widetilde{C}_i = \widetilde{A}_i^\intercal \widetilde{B}_i = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} A_j^\intercal B_k x_i^{\alpha j + \beta k}.$$

- To recover the mn products $A_j^\intercal B_k$, we need to choose (α, β) such that no two products share the same monomial $x^{\alpha j + \beta k}$.
- In this scenario, a simple choice is $(\alpha, \beta) = (1, m)$, that is,

$$\widetilde{C}_i = \widetilde{A}_i^\intercal \widetilde{B}_i = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} A_j^\intercal B_k x_i^{j + mk}.$$

- We define now the matrix polynomial

$$h(x) = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} A_j^\intercal B_k x^{j + mk} \in \mathbb{F}^{r' \times t'}[x].$$

Polynomial Codes

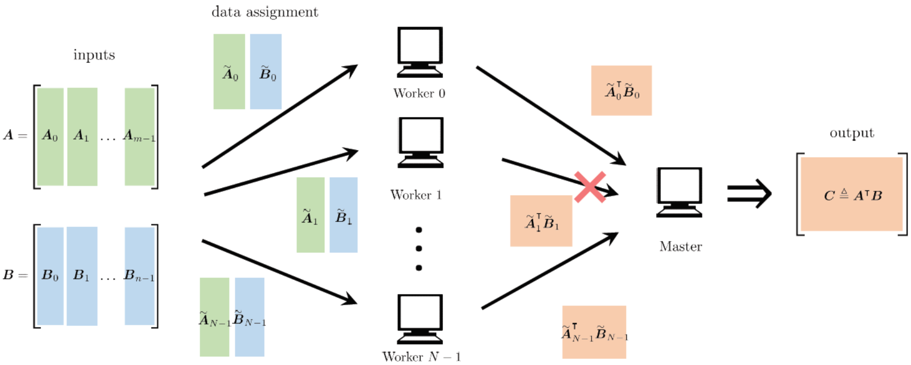
- We define now the matrix polynomial

$$h(x) = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} A_j^\intercal B_k x^{j+mk} \in \mathbb{F}^{r' \times t'}[x].$$

- Since $\deg(h(x)) = mn - 1$, and the products $A_j^\intercal B_k$ appear with different monomials, then we only need to collect outputs from mn workers and apply Lagrange interpolation.
- The number of workers N is arbitrary with $N \geq mn$.
- Hence the polynomial code can tolerate up to $N - mn$ stragglers.

Q. Yu, M. Maddah-Ali and S. Avestimehr.
Polynomial codes: an optimal design for high-dimensional coded matrix multiplication.
Advances in Neural Information Processing Systems, 30, 2017.

Polynomial Codes



MatDot Codes

- Polynomial codes are optimal only for some metrics (more later).
- Consider now $A, B \in \mathbb{F}^{N \times N}$ and let's compute $C = AB$.
- As a toy example, for polynomial codes we subdivide A and B as

$$A = \begin{pmatrix} A_0 \\ A_1 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} B_0 & B_1 \end{pmatrix}.$$

- The product $C = AB$ can be decomposed as

$$C = AB = \begin{pmatrix} A_0 B_0 & A_0 B_1 \\ A_1 B_0 & A_1 B_1 \end{pmatrix}.$$

- As we have seen, for these codes we need to recover (by interpolation) the polynomial

$$h(x) = A_0 B_0 + A_1 B_0 x + A_0 B_1 x^2 + A_1 B_1 x^3 \in \mathbb{F}^{N/2 \times N/2}[x].$$

MatDot Codes

- Polynomial codes are optimal only for some metrics (more later).
- Consider now $A, B \in \mathbb{F}^{N \times N}$ and let's compute $C = AB$.
- As a toy example, for MatDot codes we subdivide A and B as

$$A = \begin{pmatrix} A_0 & A_1 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} B_0 \\ B_1 \end{pmatrix}.$$

- With this decomposition, we simply have

$$C = AB = A_0 B_0 + A_1 B_1.$$

- If we set $p_A(x) = A_0 + A_1 x$ and $p_B(x) = B_0 x + B_1$, then

$$h(x) = p_A(x) p_B(x) = A_0 B_1 + (A_0 B_0 + A_1 B_1) x + A_1 B_0 x^2.$$

- We can recover $AB = A_0 B_0 + A_1 B_1$ from any 3 workers by collecting 3 evaluations $h(x_{i_1})$, $h(x_{i_2})$ and $h(x_{i_3})$.

MatDot Codes

- In general, for MatDot codes we subdivide

$$A = \begin{pmatrix} A_0 & A_1 & \dots & A_{m-1} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} B_0 \\ \vdots \\ B_{m-1} \end{pmatrix},$$

  where $m \mid N$, $A, B \in \mathbb{F}^{N \times N}$, $A_i \in \mathbb{F}^{N \times N/m}$, $B_j \in \mathbb{F}^{N/m \times N}$.
- We choose distinct $x_1, x_2, \dots, x_P \in \mathbb{F}$, and set

$$p_A(x) = \sum_{i=0}^{m-1} A_i x^i \quad \text{and} \quad p_B(x) = \sum_{j=0}^{m-1} B_j x^{m-1-j}.$$

- The *i*th worker obtains $p_A(x_i)$ and $p_B(x_i)$ and computes $h(x_i) = p_A(x_i) p_B(x_i)$, for $i = 1, 2, \dots, P$.
- We have that $AB = \sum_{j=0}^{m-1} A_j B_j$ is the coefficient of x^{m-1} in $h(x)$.
- Since $\deg(h(x)) \leq 2m - 2$, we only need to collect the evaluations of $2m - 1$ (out of P) workers.

PolyDot Codes

- We can also obtain hybrid solutions: PolyDot codes.
- Toy example: We split $A, B \in \mathbb{F}^{N \times N}$ as

$$A = \begin{pmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} B_{0,0} & B_{0,1} \\ B_{1,0} & B_{1,1} \end{pmatrix}.$$

- We have that

$$AB = \begin{pmatrix} A_{0,0}B_{0,0} + A_{0,1}B_{1,0} & A_{0,0}B_{0,1} + A_{0,1}B_{1,1} \\ A_{1,0}B_{0,0} + A_{1,1}B_{1,0} & A_{1,0}B_{0,1} + A_{1,1}B_{1,1} \end{pmatrix}.$$

- Set $p_A(x) = A_{0,0} + A_{1,0}x + A_{0,1}x^2 + A_{1,1}x^3$ and $p_B(x) = B_{0,0}x^2 + B_{1,0} + B_{0,1}x^8 + B_{1,1}x^6$.
- The 4 block components of AB are the coefficients of

$$x^2, \quad x^8, \quad x^3 \quad \text{and} \quad x^9.$$

- We may recover AB from 4 evaluations of $h(x) = p_A(x)p_B(x)$.

Hybrid Solution: PolyDot Codes

- In general: We split $A, B \in \mathbb{F}^{N \times N}$ as

$$A = \begin{pmatrix} A_{0,0} & \cdots & A_{0,s-1} \\ \vdots & \ddots & \vdots \\ A_{t-1,0} & \cdots & A_{t-1,s-1} \end{pmatrix}, \quad B = \begin{pmatrix} B_{0,0} & \cdots & B_{0,s-1} \\ \vdots & \ddots & \vdots \\ B_{t-1,0} & \cdots & B_{t-1,s-1} \end{pmatrix}.$$

- We define

$$p_A(x) = \sum_{i=0}^{t-1} \sum_{j=0}^{s-1} A_{i,j} x^{i+tj},$$

$$p_A(x) = \sum_{k=0}^{s-1} \sum_{l=0}^{t-1} B_{k,l} x^{t(s-1-k)+t(2s-1)l}.$$

- If $h(x) = p_A(x) p_B(x)$, then

$$h(x) = \sum_{i,j,k,l} A_{i,j} B_{k,l} x^{i+t(s-1+j-k)+t(2s-1)l}.$$

Hybrid Solution: PolyDot Codes

- If $h(x) = p_A(x)p_B(x)$, then

$$h(x) = \sum_{i,j,k,l} A_{i,j} B_{k,l} x^{i+t(s-1+j-k)+t(2s-1)l}.$$

- It can be shown that for different pairs of (i, jk, l) we get different powers of x.
- For the powers such that $j - k = 0$, we have the term

$$\left(\sum_{k=0}^{s-1} A_{i,k} B_{k,l} \right) x^{i+t(s-1)+t(2s-1)l} = C_{i,l} x^{i+t(s-1)+t(2s-1)l}.$$

- Notice that

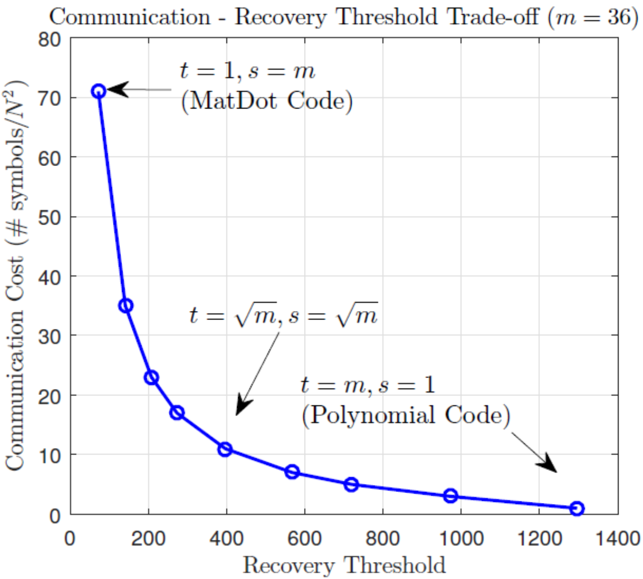
$$\deg(h(x)) \leq t - 1 + 2t(s-1) + t(2s-1)(t-1) = t^2(2s-1).$$

- Thus we need $t^2(2s-1)$ responses from the workers.

Communication - Recovery Trade-Off

- We have divided $A, B \in \mathbb{F}^{N \times N}$ into $m := st$ submatrices. Each worker stores $2N^2/m$ symbols in $\mathbb{F}$.
- Keeping storage cost constant, i.e. $m = st$ constant, the recovery threshold
$$t^2(2s-1) = m^2 \cdot \frac{2s-1}{s^2}$$
decreases as s increases, i.e., increases as t increases.
- In terms of communication cost, the master node sends $\mathcal{O}(N^2/m)$ symbols to each worker, and each worker sends $\mathcal{O}(N^2/t^2)$ symbols to the fusion node.
- Since we collect outputs from $t^2(2s-1)$ workers, the total communication cost from the workers to the fusion node is
$$\mathcal{O}\left(t^2(2s-1) \cdot \frac{N^2}{t^2}\right) = \mathcal{O}(N^2(2s-1)),$$
- which increases as s increases, i.e., decreases as t increases.

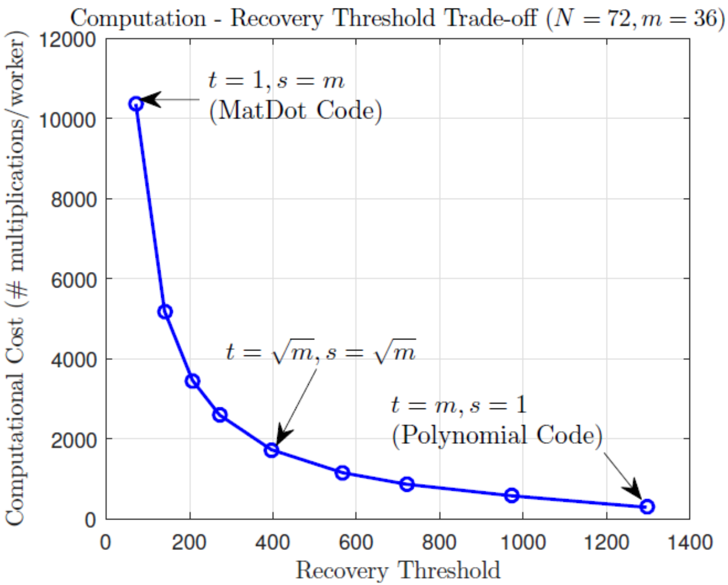
Communication - Recovery Trade-Off



Computation - Recovery Trade-Off

- We have divided $A, B \in \mathbb{F}^{N \times N}$ into $m := st$ submatrices. Each worker stores $2N^2/m$ symbols in $\mathbb{F}$.
- Keeping storage cost constant, i.e. $m = st$ constant, the recovery threshold
$$t^2(2s-1) = m^2 \cdot \frac{2s-1}{s^2}$$
decreases as s increases, i.e., increases as t increases.
- In terms of computation cost, each worker computes the product of $N/t \times N/s$ and $N/s \times N/t$ matrices, which has a computational cost (over $\mathbb{F}$) of
$$\mathcal{O}\left(\frac{N^3}{st^2}\right) = \mathcal{O}\left(\frac{N^3}{m^2} \cdot s\right),$$
- which increases as s increases, i.e., decreases as t increases.
- For the decoding to be negligible in comparison, we need
$$m^2 t^2 = \frac{m^4}{s^2} = o(N).$$

Computation - Recovery Trade-Off



PolyDot Codes

- MatDot codes and PolyDot codes, together with the previous trade-offs, were introduced in

  S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe and P. Grover.
  On the optimal recovery threshold of coded matrix multiplication.
  IEEE Trans. Info. Theory, 66(1):278–301, 2019.

- They also give upper bounds and show that PolyDot codes always attain the bounds.

- For $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$, the number of workers is unrestricted, the main issues have to do with numerical stability:

  M. Fahim and V. Cadambe.
  Numerically stable polynomially coded computing.
  IEEE Trans. Info. Theory, 67(5):2758–2785, 2021.

PolyDot Codes

- For $\mathbb{F} = \mathbb{F}_q$, numerical stability is not a problem, but the q may be.
- We need the number of workers to satisfy $P \leq q$.
- But the recovery threshold also needs to satisfy

$$t^2(2s-1) \leq P$$

- Hence PolyDot codes require

$$q \geq t^2(2s-1) = m^2 \cdot \frac{2s-1}{s^2}.$$

- Alternatives: 1) Using ideas from Algebraic-Geometry codes? Problem: Degrees. Maybe we need to choose algebraic functions appropriately.
- Alternatives: 2) Using polynomials in several variables and/or certain evaluation points.
- Alternatives: 3) Using ideas from subfield subcodes.

Secure Distributed Matrix Multiplication

- We now consider the problem of multiplying two matrices in a secure way.
- We want to multiply A and B using N workers in a way that no T of them can obtain any information about A or B (in an IT sense).
- In this scenario, we assume all N workers are responsive (on time and correct) (honest but curious).
- For this problem, it is usual to consider as performance metric the download rate, which is inverse to the communication cost.
- Recall that for PolyDot codes the communication cost was

$$\mathcal{O}(N^2(2s-1)),$$

  which is minimum for polynomial codes, $s=1$.
- For this reason, most works on SDMM follow the same matrix subdivision as polynomial codes.

Secure Distributed Matrix Multiplication

We start with the matrix subdivision of polynomial codes

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_K \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} B_1 & B_2 & \ldots & B_L \end{pmatrix},$$

so that

$$C = AB = \begin{pmatrix} A_1 B_1 & A_1 B_2 & \ldots & A_1 B_L \\ A_2 B_1 & A_2 B_2 & \ldots & A_2 B_L \\ \vdots & \vdots & \ddots & \vdots \\ A_K B_1 & A_K B_2 & \ldots & A_K B_L \end{pmatrix}.$$

Secure Distributed Matrix Multiplication

- We fix T such that no T workers will be able to obtain any information about A or B.
- Fix degree sequences

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{K+T}) \quad \text{and} \quad \beta = (\beta_1, \beta_2, \ldots, \beta_{L+T}).$$

- Generate random matrices $R_1, R_2, \ldots, R_T$ and $S_1, S_2, \ldots, S_T$, of appropriate sizes, and define

$$f(x) = \sum_{k=1}^{K} A_k x^{\alpha_k} + \sum_{t=1}^{T} R_t x^{\alpha_{K+t}},$$
$$g(x) = \sum_{\ell=1}^{L} B_\ell x^{\beta_\ell} + \sum_{t=1}^{T} S_t x^{\beta_{L+t}}.$$

Secure Distributed Matrix Multiplication

- Fix also distinct $a_1, a_2, \ldots, a_N \in \mathbb{F}$. We send $f(a_i)$ and $g(a_i)$ to the ith worker, who computes

$$h(a_i) = f(a_i)g(a_i).$$

- As in Shamir's scheme, we want to recover all products $A_k B_\ell$ from $h(a_1), h(a_2), \ldots, h(a_N)$,
- while no information about A and B (i.e., the matrices A_k and B_ℓ) is leaked from any T evaluations of $h(x)$:

$$I(f(a_{i_1}), g(a_{i_1}), \ldots, f(a_{i_T}), g(a_{i_T}); A, B) = 0.$$

- The download rate of the scheme is defined as

$$\mathcal{R} = \frac{KL}{N}.$$

Secure Distributed Matrix Multiplication

- We define the degree table as

$$\alpha \oplus \beta = \begin{pmatrix} \alpha_1 + \beta_1 & \cdots & \alpha_1 + \beta_{L+T} \\ \vdots & \ddots & \vdots \\ \alpha_{K+T} + \beta_1 & \cdots & \alpha_{K+T} + \beta_{L+T} \end{pmatrix}.$$

- The scheme satisfies the recovery and secrecy conditions iff
  - $\alpha_k + \beta_\ell \neq \alpha_{k'} + \beta_{\ell'}$, for all $(k, \ell) \in [K] \times [L]$ and all $(k', L') \in [K+T] \times [L+T]$.
  - $\alpha_{K+t} \neq \alpha_{K+t'}$ and $\beta_{L+t} \neq \beta_{L+t'}$, for all $t \neq t' \in [T]$.

	β_1	$\cdots$	β_L	β_{L+1}	$\cdots$	β_{L+T}
α_1	$\alpha_1+\beta_1$	$\cdots$	$\alpha_1+\beta_L$	$\alpha_1+\beta_{L+1}$	$\cdots$	$\alpha_1+\beta_{L+T}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
α_K	$\alpha_K+\beta_1$	$\cdots$	$\alpha_K+\beta_L$	$\alpha_K+\beta_{L+1}$	$\cdots$	$\alpha_K+\beta_{L+T}$
α_{K+1}	$\alpha_{K+1}+\beta_1$	$\cdots$	$\alpha_{K+1}+\beta_L$	$\alpha_{K+1}+\beta_{L+1}$	$\cdots$	$\alpha_{K+1}+\beta_{L+T}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
α_{K+T}	$\alpha_{K+T}+\beta_1$	$\cdots$	$\alpha_{K+T}+\beta_L$	$\alpha_{K+T}+\beta_{L+1}$	$\cdots$	$\alpha_{K+T}+\beta_{L+T}$

Secure Distributed Matrix Multiplication

A first valid choice of α and β is

$$\alpha_k = \begin{cases} k-1 & \text{if } 1 \leq k \leq K, \\ KL+t-1 & \text{if } k = K+t \text{ and } 1 \leq t \leq T, \end{cases}$$

$$\beta_\ell = \begin{cases} K(\ell-1) & \text{if } 1 \leq \ell \leq L, \\ KL+t-1 & \text{if } \ell = L+t \text{ and } 1 \leq t \leq T, \end{cases}$$

if $L \leq K$, and

$$\alpha_\ell = \begin{cases} K(\ell-1) & \text{if } 1 \leq \ell \leq L, \\ KL+t-1 & \text{if } \ell = L+t \text{ and } 1 \leq t \leq T, \end{cases}$$

$$\beta_k = \begin{cases} k-1 & \text{if } 1 \leq k \leq K, \\ KL+t-1 & \text{if } k = K+t \text{ and } 1 \leq t \leq T, \end{cases}$$

if $K < L$.

Secure Distributed Matrix Multiplication

$$\alpha_k = \begin{cases} k-1 & \text{if } 1 \leq k \leq K, \\ KL+t-1 & \text{if } k = K+t \text{ and } 1 \leq t \leq T, \end{cases}$$

$$\beta_\ell = \begin{cases} K(\ell-1) & \text{if } 1 \leq \ell \leq L, \\ KL+t-1 & \text{if } \ell = L+t \text{ and } 1 \leq t \leq T, \end{cases}$$

if $L \leq K$.

	$\beta_1 = 0$	$\cdots$	$\beta_L = K(L-1)$	$\beta_{L+1} = KL$	$\beta_{L+2} = KL+1$	$\cdots$	$\beta_{L+T} = KL+T-1$
$\alpha_1 = 0$	0	$\cdots$	$K(L-1)$	KL	$KL+1$	$\cdots$	$KL+T-1$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\alpha_K = K-1$	$K-1$	$\cdots$	$KL-1$	$KL+K-1$	$KL+K$	$\cdots$	$KL+K+T-2$
$\alpha_{K+1} = KL$	KL	$\cdots$	$2KL-K$	$2KL$	$2KL+1$	$\cdots$	$2KL+T-1$
$\alpha_{K+2} = KL+1$	$KL+1$	$\cdots$	$2KL-K+1$	$2KL+1$	$2KL+2$	$\cdots$	$2KL+T$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\alpha_{K+T} = KL+T-1$	$KL+T-1$	$\cdots$	$2KL-K+T-1$	$2KL+T-1$	$2KL+T$	$\cdots$	$2KL+2T-2$

Secure Distributed Matrix Multiplication

- The number of workers is given by

$$N = \begin{cases} (K+T)(L+1)-1 & \text{if } T < K, \\ 2KL+2T-1 & \text{if } T \geq K, \end{cases}$$

  if $L \leq K$, and

- $$N = \begin{cases} (L+T)(K+1)-1 & \text{if } T < L, \\ 2KL+2T-1 & \text{if } T \geq L, \end{cases}$$

  if $L < K$.

- The rate is given by $\mathcal{R} = KL/N$ with N as above.

- They all satisfy $N \leq (K+T)(L+1)-1$, so

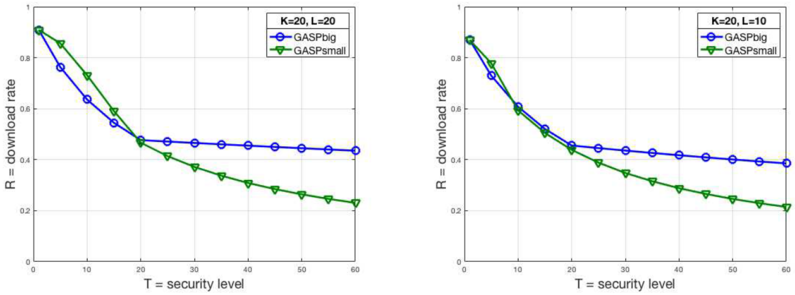
$$\mathcal{R} \geq \frac{KL}{(K+T)(L+1)-1}.$$

- This scheme was introduced in

  R. D'Oliveira, S. El Rouayheb and D. Karpuk.
  GASP codes for secure distributed matrix multiplication.
  IEEE Trans. Info. Theory, 66(7):4038–4050, 2020.

- They also provide another scheme that is slightly better for

$$T < \max\{K, L\}.$$



Secure Distributed Matrix Multiplication

- As in the case of recovery, for $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$, the main problem seems to be numerical stability.
- For $\mathbb{F} = \mathbb{F}_q$, since the number of workers is about $N \cong (K + T)(L + 1) - 1$, therefore we need

$$q \geq (K + T)(L + 1) - 1.$$

- Alternative solutions may require algebraic-geometry codes, multivariate polynomials, etc.
- The previous work only considered communication cost (i.e. download rate) as performance metric.
- Finding good codes for other metrics, such as recovery threshold (i.e. PolyDot codes) seems to be an open problem.

Related Problems

- Private computation consists in distributing a computation among workers while maintaining private the computation itself.

N. Raviv and D. Karpuk.
Private polynomial computation from Lagrange encoding.
IEEE Trans. Info. Forensics and Security, 15:553–563, 2019.

- Another approach to distributed matrix multiplication is using partial results from all workers:

N. Ferdinand and S. Draper.
Anytime stochastic gradient descent: A time to hear from all the workers.
56th Allerton Conf. Comm. Control Comp., 552–559, 2018.

S. Kianidehkordi, N. Ferdinand and S. Draper.
Hierarchical coded matrix multiplication.
IEEE Trans. Info. Theory, 67(2):726–754, 2020.

Related Problems

- There are works considering simultaneously recovery, security and privacy.

  Q. Yu, S. Li, N. Raviv, S. Kalan, M. Soltanolkotabi, S. Avestimehr.
  Lagrange coded computing: Optimal design for resiliency, security, and privacy.
  22nd Int. Conf. Artif. Intel. Stat., 1215–1225, 2019.

- There are coding solutions to straggler mitigation for specific Machine Learning algorithms, such as gradient descent:

  N. Raviv, I. Tamo, R. Tandon and A. Dimakis.
  Gradient coding from cyclic MDS codes and expander graphs.
  IEEE Trans. Info. Theory, 66(12):7475–7489, 2020.

Conclusion

Thank you for your attention.