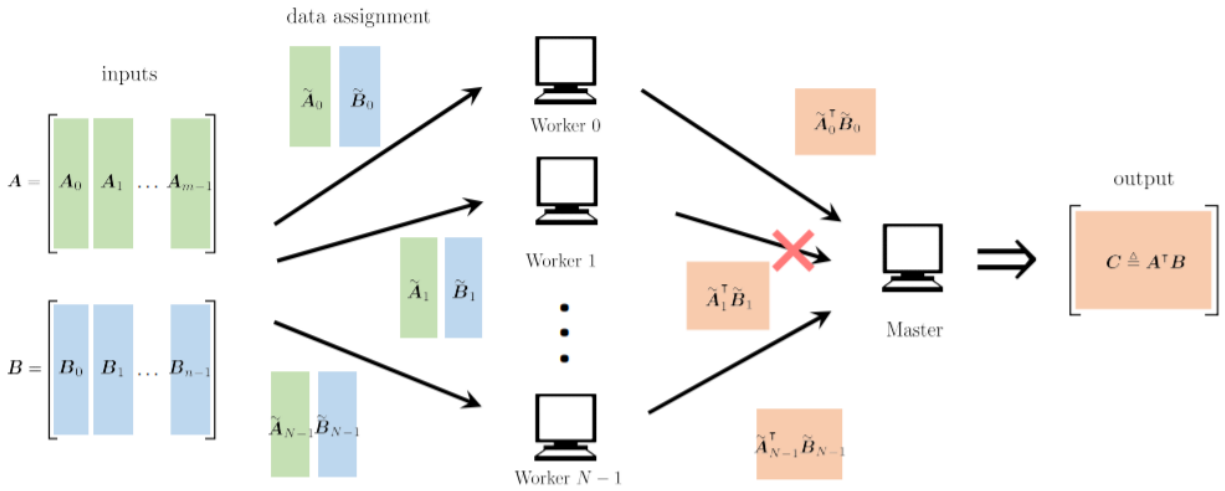
Coded Computing for Straggler Mitigation, Security and Privacy

Param Rathour 190070049, Anupam Nayak 19D070010

Video available here:
Presentation available here (fixed typos)

Fathima Zarin Faizal (180070018):
Overall great presentation, had a few doubts:
I didn't quite get proof I so would be nice if you could elaborate. What exactly are x_is and what does each worker calculate? Is it like you predetermine xis and each worker returns \overline{C_i} so we can interpolate and get the coefficients of h? But each worker stores only a fraction of A and B and the computation for \overline{A_i} requires more than that? (slide 9-10)

A) Each worker is supposed to calculate the product of \overline{A_i} transpose and \overline{B_i.}Each worker stores \overline{A_i} and \overline{B_i}(assigned to it by the master which implements the computation strategy(Slide 5) to obtain them and sends these matrices to the worker - polynomial computation strategy is defined in slide 9) which is indeed calculated from A_i and B_i as according to the polynomial computation strategy. Storing 1/m fraction of A meant storing a matrix whose dimension is s\times r/m which is 1/m times the original.(Linear combination of A_i). Same goes for B_i. I guess the graphic below would be more helpful here rather than the one used in slides where the matrices were further split row wise



x_i are the points used in polynomial interpolation and yes they are distinct and generally pre determined

Also, how are we guaranteed that the minimum recovery threshold for polynomial evaluation is the value given? We just detailed an encoding/decoding scheme to achieve that value (K*) I think (?) or am I missing something (slide 15)

You are right about slide 15. The proof given is for optimality of LCC and the proof for the minimum recovery threshold was skipped (as mentioned in the video). I will sketch the proof here for the interested,

- First, this theorem is proved for a special case when f is a multilinear function (f is linear in each variable when rest of the variables are fixed). Now, the main idea is to show that for any K < K*, any strategy will fail for some computation of f.
- Next, this result can be generalized to arbitrary polynomials f, by constructing a non-zero multilinear polynomial f' of same degree d (= deg f). f' is defined as linear combination of functions which are a composition of a linear map and f:
  $f'(Z_1,...,Z_d) = \sum_{S \subseteq [d]} (-1)^{|S|} \cdot f(\Sigma_{j \in S} Z_j)$ for any $\{Z_j\}_{j \in [d]} \in V^d$
  Now, we use the result of multilinear functions to get a lower bound on the recovery threshold of f
  K* of f >= K* for f' = (K-1) deg f + 1 (as f, f' are of same degree)

Each step is highly involved but I hope you get the idea :)
Do check out appendix E of the LCC paper for more details

Thank you for the feedback:)

Ujjwal Kalra (18D070031):
Really great presentation and use of graphics to explain, especially the one on Secure and Private MultiParty Computing. Your slides actually gave me an appearance of slides by our institute's Professors.
Neatly defining the terminologies helped in understanding effectively.

Thank you for the feedback:)

Saketika Chekuri (190070054):
Nice presentation, but I found the notation a little confusing. Could you please explain on slide 9 what the x's and \bar {A_i} represent?

x_i are the points used in polynomial interpolation, they are distinct and generally pre determined. It is replaced by x in the next slide for representing one instance of these points nonetheless x_i notation would also do

A_i, B_i represent the columns of matrices A, B respectively. Each worker is supposed to calculate the product of \overline{A_i} transpose and \overline{B_i.}Each worker stores \overline{A_i} and \overline{B_i}(assigned to it by the master which implements the computation strategy(Slide 5) to obtain them and sends these matrices to the worker - polynomial computation strategy is defined in slide 9) which is indeed calculated from A_i and B_i as according to the polynomial computation strategy.

Also, on slide 11, it's been given that H(c) = rt*log q. How are we sure that the elements are independent for the equality to hold? In general, won't it be <= rt*log q?

> Yeah I missed a small yet important detail that gives this equality in the presentation. It was corrected in the updated slides. The fixed matrix A (dim s \times r) needs to be tall and **full rank**. Thus the. Thus the distribution of C = A^T B will be uniform over F_q^{r \times t) And thus the entropy would be rt log q
>
> Thanks for the feedback :)

Varad Mane(18D070034):

1. What's meant by a bilinear complex code, is it a random linear code, and how it the threshold constant for a random code?

> This is another computation strategy like the polynomial, 1d MDS computation strategy used for coded matrix multiplications, couldn't cover due to lack of time
>
> However these were our main references for the same may be helpful
> The bilinear complexity and practical algorithms for matrix multiplication
> Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding

2. Is there any general approach to choose alpha, beta(slide 9)?

> α and β are chosen such that no two terms end up having the same exponent of x in the expression below

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

.

> One such choice is α = 1 and β = m

3. Nice use of graphs, good presentation

> Thanks for the feedback :)

Yash Dixit(180260043):

How do we know that the sum of the (rt/mn)logq's will be rtlogq? Isnt it possible that 2 workers are working with some redundant information, so the information they produce might have some overlap?

> The exact proof has been skipped here and only a brief sketch was provided, In the full proof a cut set bound is used to ensure this. But the intuition here is that we would need at least mn workers with non redundant information to obtain rt log q information. Thus K* \geq mn which is achievable using polynomial code.

Thanks for the feedback :)

Chinmay Bharti (18d070043):
The presentation was very nice and well crafted. The use of graphs and diagrams really helped to build the intuition, I only have one small doubt. In the proof 1 you have stated that we have to choose alpha and beta such that no 2 terms have the same power of x. Can you please explain why is that so and what will happen otherwise?

α and β are chosen such that no two terms end up having the same exponent of x in the expression below

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

.

One such choice is α = 1 and β = m. If this isn't the case and we end up having terms with common exponents i.e. the polynomial will have terms like

(A_i B_j +A_k B_l + …. A_p B_q)x^k we would have to figure out ways of getting back A_i B_j , A_k B_l etc (We need all the sub products to decode back the original product C) from their sum after polynomial interpolation which will only give the coefficient ie the sum which I don't think is possible by design here.

Thanks for the feedback :)

Shreyashree Satyen (17B030012):
Wonderful presentation. Do expand on implementations outside of what was discussed in the video (esp instrumentation engineering etc)]

Thanks for the feedback :)

Also, how would you explain this to a teenager?

This is all the trouble my partner would go through

"The best I can do is to provide intuition about the benefits of coding and distributed computing. Say you want to multiply a matrix A with a vector b and you have 2 workers. A naive method will be to **distribute** A row-wise (A_1 and A_2) and give both these computations to 2 workers ($A_1 \cdot b$ and $A_2 \cdot b$) but if one of the workers fail we can't our result back. If we add a third worker which needs to compute (A_1 + A_2) · b

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix} \quad \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_1 + \mathbf{A}_2 \end{bmatrix}$$

> And with simple **coding**, we encode A as shown above. Now, if any 1 worker fails we can retrieve back $A \cdot b$ (**decoding**) by using the computations of the other 2 workers. In this way, we get protection from 1 straggler."

If they understand matrix multiplications, polynomial evaluations, the meaning of computation, privacy, the basics of ECC, I would go with pretty much the same thing I have done here along with some details about MDS codes, Reed Solomon codes if necessary.
If that's not the case I highly doubt that I would even try :)