Dual-Entangled Polynomial Code: Three-Dimensional Coding for Distributed Matrix Multiplication

Pedro Soto, Jun Li, Xiaodi Fan
Florida International University

FIU Computing & Information Sciences
Florida International University

Matrix Multiplication

- Matrix multiplication is a fundamental building block in various machine learning algorithms.
- When the matrix comes from a large dataset, the multiplication will be split into smaller multiplications of submatrices on different nodes.

Matrix Multiplication

- Matrix multiplication is a fundamental building block in various machine learning algorithms.
- When the matrix comes from a large dataset, the multiplication will be split into smaller multiplications of submatrices on different nodes.

$$A \cdot B$$

Matrix Multiplication

- Matrix multiplication is a fundamental building block in various machine learning algorithms.
- When the matrix comes from a large dataset, the multiplication will be split into smaller multiplications of submatrices on different nodes.

$$A \cdot B = \begin{bmatrix} A_0 \\ A_1 \end{bmatrix} \cdot B$$

Matrix Multiplication

- Matrix multiplication is a fundamental building block in various machine learning algorithms.
- When the matrix comes from a large dataset, the multiplication will be split into smaller multiplications of submatrices on different nodes.

$$A \cdot B = \begin{bmatrix} A_0 \\ A_1 \end{bmatrix} \cdot B$$

worker

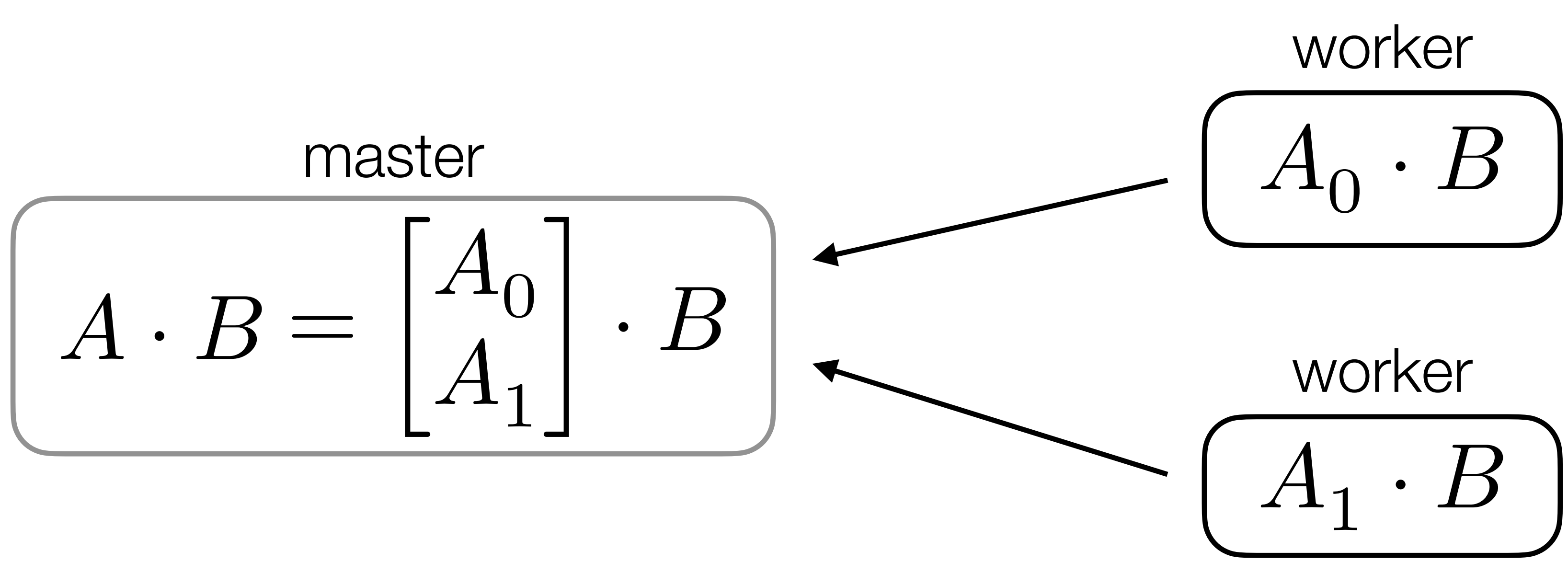
$$A_0 \cdot B$$

worker

$$A_1 \cdot B$$

Matrix Multiplication

- Matrix multiplication is a fundamental building block in various machine learning algorithms.
- When the matrix comes from a large dataset, the multiplication will be split into smaller multiplications of submatrices on different nodes.

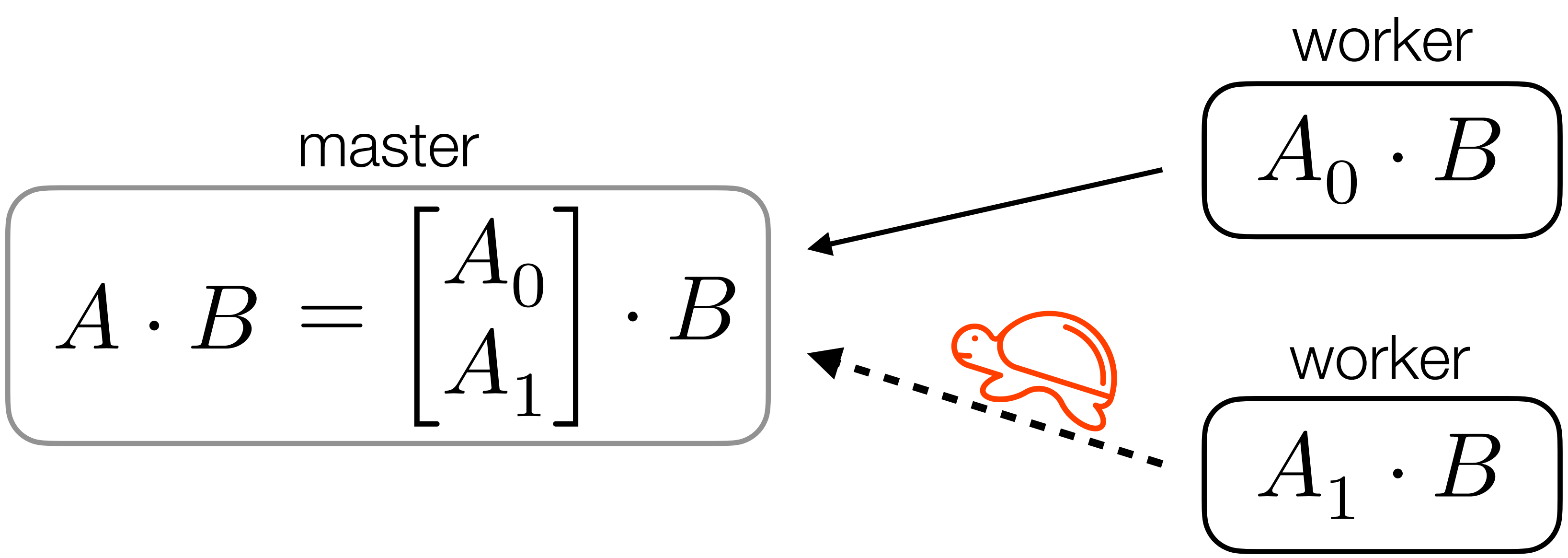


Straggler

- Stragglers are common in distributed systems, due to load imbalance, resource contention, *etc*.

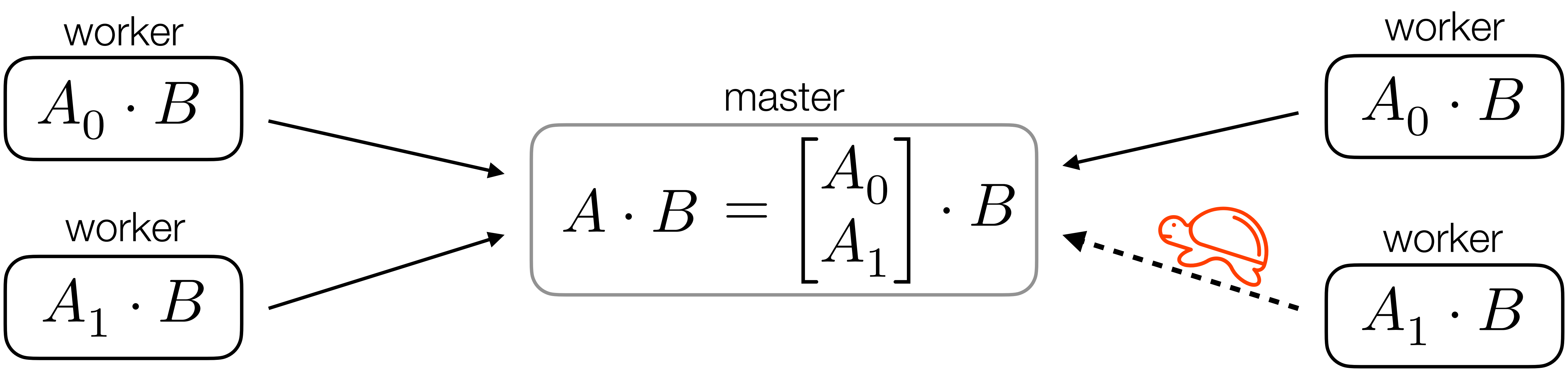
Straggler

- Stragglers are common in distributed systems, due to load imbalance, resource contention, *etc*.



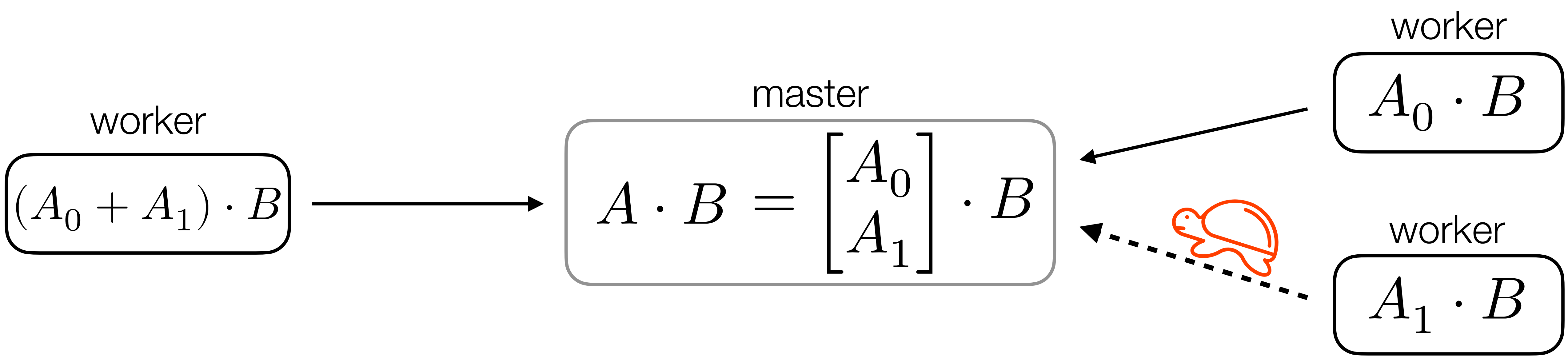
Straggler

- Stragglers are common in distributed systems, due to load imbalance, resource contention, *etc*.



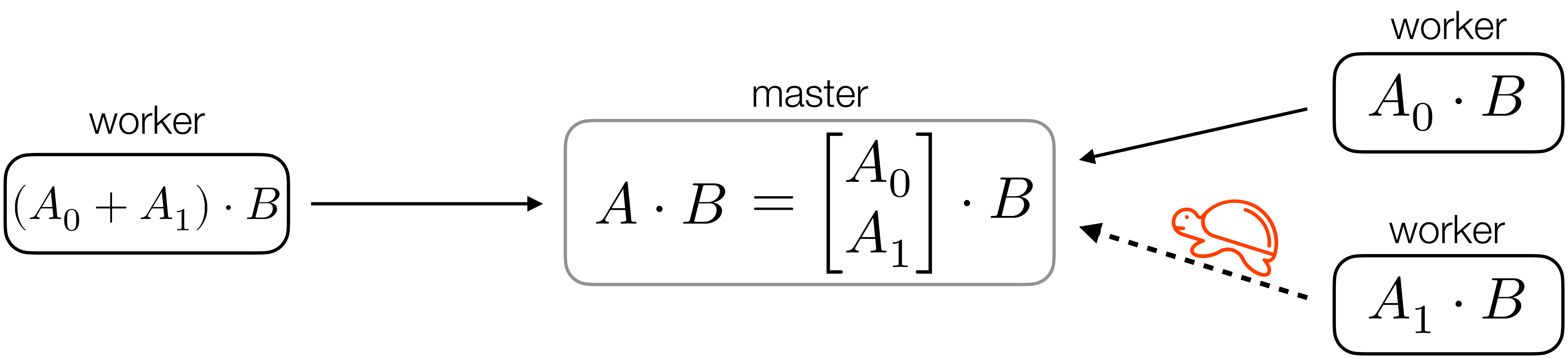
Straggler

- Stragglers are common in distributed systems, due to load imbalance, resource contention, *etc*.



Straggler

- Stragglers are common in distributed systems, due to load imbalance, resource contention, *etc*.



- Coded matrix multiplication can tolerate stragglers with fewer tasks.

Related Works

coding	matrix partition	recovery threshold
1D [Lee et al., Trans. IT, 2018]	$\begin{bmatrix} A_0 \\ \vdots \\ A_{x-1} \end{bmatrix} \cdot B$	*x*
2D [Yu et al., NIPS 2017]	$\begin{bmatrix} A_0 \\ \vdots \\ A_{x-1} \end{bmatrix} \cdot \begin{bmatrix} B_0 & \cdots & B_{y-1} \end{bmatrix}$	*xy*
3D [Yu et al., ISIT 2018]	$\begin{bmatrix} A_{0,0} & \cdots & A_{0,z-1} \\ \vdots & \ddots & \vdots \\ A_{x-1,0} & \cdots & A_{x-1,z-1} \end{bmatrix} \cdot \begin{bmatrix} B_{0,0} & \cdots & B_{0,y-1} \\ \vdots & \ddots & \vdots \\ B_{z-1,0} & \cdots & B_{z-1,y-1} \end{bmatrix}$	*xyz+z-1*

Entangled Polynomial Code

- Entangled Polynomial (EP) code [Yu et al., ISIT 2018] is the state-of-the-art three-dimensional coding.
- For example, if $A = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$ and $B = \begin{bmatrix} B_{0,0} & B_{0,1} \\ B_{1,0} & B_{1,1} \end{bmatrix}$, i.e., $x=y=z=2$, a task will be:

$$((A_{0,0}\delta^0 + A_{0,1}\delta^1)\delta^0 + (A_{1,0}\delta^0 + A_{1,1}\delta^1)\delta^4) \times ((B_{0,0}\delta^1 + B_{1,0}\delta^0)\delta^0 + (B_{0,1}\delta^1 + B_{1,1}\delta^0)\delta^2)$$

Entangled Polynomial Code

- Entangled Polynomial (EP) code [Yu et al., ISIT 2018] is the state-of-the-art three-dimensional coding.
- For example, if $A = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$ and $B = \begin{bmatrix} B_{0,0} & B_{0,1} \\ B_{1,0} & B_{1,1} \end{bmatrix}$, i.e., x=y=z=2, a task will be:

$$(\underbrace{(A_{0,0}\delta^0 + A_{0,1}\delta^1)}_{\tilde{A}_0}\delta^0 + \underbrace{(A_{1,0}\delta^0 + A_{1,1}\delta^1)}_{\tilde{A}_1}\delta^4) \times (\underbrace{(B_{0,0}\delta^1 + B_{1,0}\delta^0)}_{\tilde{B}_0}\delta^0 + \underbrace{(B_{0,1}\delta^1 + B_{1,1}\delta^0)}_{\tilde{B}_1}\delta^2)$$

Entangled Polynomial Code

- Entangled Polynomial (EP) code [Yu et al., ISIT 2018] is the state-of-the-art three-dimensional coding.
- For example, if $A = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$ and $B = \begin{bmatrix} B_{0,0} & B_{0,1} \\ B_{1,0} & B_{1,1} \end{bmatrix}$, i.e., $x=y=z=2$, a task will be:

$$\Big(\overset{\tilde{A}_0}{\boxed{(A_{0,0}\delta^0 + A_{0,1}\delta^1)}}\delta^0 + \overset{\tilde{A}_1}{\boxed{(A_{1,0}\delta^0 + A_{1,1}\delta^1)}}\delta^4\Big) \times \Big(\overset{\tilde{B}_0}{\boxed{(B_{0,0}\delta^1 + B_{1,0}\delta^0)}}\delta^0 + \overset{\tilde{B}_1}{\boxed{(B_{0,1}\delta^1 + B_{1,1}\delta^0)}}\delta^2\Big)$$

$\tilde{A}_0\tilde{B}_0\delta^0$	$\rightarrow$	$A_{0,0}B_{1,0}\delta^0 + \left(A_{0,0}B_{0,0} + A_{0,1}B_{1,0}\right)\delta^1 + A_{0,1}B_{0,0}\delta^2$
$\tilde{A}_0\tilde{B}_1\delta^2$	$\rightarrow$	$A_{0,0}B_{1,1}\delta^2 + \left(A_{0,0}B_{0,1} + A_{0,1}B_{1,1}\right)\delta^3 + A_{0,1}B_{0,1}\delta^4$
$\tilde{A}_1\tilde{B}_0\delta^4$	$\rightarrow$	$A_{1,0}B_{1,0}\delta^4 + \left(A_{1,0}B_{0,0} + A_{1,1}B_{1,0}\right)\delta^5 + A_{1,1}B_{0,0}\delta^6$
$\tilde{A}_1\tilde{B}_1\delta^6$	$\rightarrow$	$A_{1,0}B_{1,1}\delta^6 + \left(A_{1,0}B_{0,1} + A_{1,1}B_{1,1}\right)\delta^7 + A_{1,1}B_{0,1}\delta^8$

Motivation

- As the input matrices are divided smaller and smaller, the number of results uploaded and decoded at the master also increase.
  - the master’s incoming traffic becomes congested, or
  - the master is overwhelmed by the decoding complexity
- Dual entangled polynomial codes allows a tradeoff between *computation* and *communication/decoding overhead*.

Dual Entangled Polynomial Code

- Dual Entangled Polynomial code (DEP) doubles the computational complexity with two multiplications, lowering the recovery threshold to $\frac{3}{4}xyz + \frac{1}{2}z - 1$. For example, when $x=y=z=2$, a task will be

$$\left(A_{0,0}\delta^0 + A_{1,1}\delta^1\right) \times \left(B_{0,0}\delta^0 + B_{1,1}\delta^1 + B_{0,1}\delta^3 + B_{1,0}\delta^4\right)$$
$$+$$
$$\left(A_{1,0}\delta^0 + A_{0,1}\delta^{-1}\right)\left(B_{0,0}\delta^0 + B_{1,1}\delta^{-1} + B_{0,1}\delta^{-3} + B_{1,0}\delta^{-4}\right)\delta^5$$

Dual Entangled Polynomial Code

- Dual Entangled Polynomial code (DEP) doubles the computational complexity with two multiplications, lowering the recovery threshold to $\frac{3}{4}xyz + \frac{1}{2}z - 1$. For example, when *x*=*y*=*z*=2, a task will be

$$\left(A_{0,0}\delta^0 + A_{1,1}\delta^1\right) \times \left(B_{0,0}\delta^0 + B_{1,1}\delta^1 + B_{0,1}\delta^3 + B_{1,0}\delta^4\right)$$
$$+$$
$$\left(A_{1,0}\delta^0 + A_{0,1}\delta^{-1}\right)\left(B_{0,0}\delta^0 + B_{1,1}\delta^{-1} + B_{0,1}\delta^{-3} + B_{1,0}\delta^{-4}\right)\delta^5$$

⇩

δ^0	δ^1	δ^2	δ^3	δ^4	δ^5
$A_{0,0}B_{0,0}$ + $A_{0,1}B_{1,0}$	noise	$A_{1,1}B_{1,1}$ + $A_{1,0}B_{0,1}$	$A_{0,0}B_{0,1}$ + $A_{0,1}B_{1,1}$	noise	$A_{1,1}B_{1,0}$ + $A_{1,0}B_{0,0}$

Dual Entangled Polynomial Code

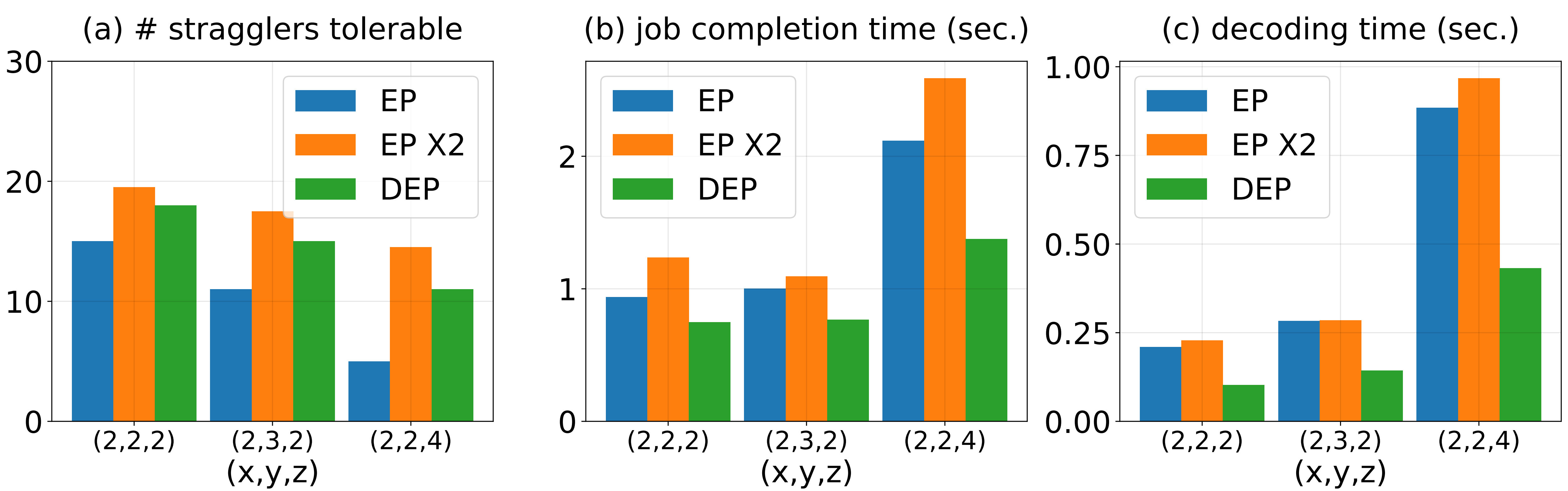
- Dual Entangled Polynomial code (DEP) doubles the computational complexity with two multiplications, lowering the recovery threshold to $\frac{3}{4}xyz + \frac{1}{2}z - 1$. For example, when x = y = z = 2, a task will be

Compared to EP codes, the recovery threshold is reduced from 9 tasks to 6 tasks.

δ^0	δ^1	δ^2	δ^3	δ^4	δ^5
$A_{0,0}B_{0,0} + A_{0,1}B_{1,0}$	noise	$A_{1,1}B_{1,1} + A_{1,0}B_{0,1}$	$A_{0,0}B_{0,1} + A_{0,1}B_{1,1}$	noise	$A_{1,1}B_{1,0} + A_{1,0}B_{0,0}$

Evaluation

- We implemented DEP codes with Open MPI, and ran the evaluation on 24 workers hosted on Microsoft Azure.
  - EP X2 runs two tasks on each worker.
  - DEP tolerates similar stragglers, while significantly saving job completion time.



Conclusion

- We propose the dual entangled polynomial code, another three-dimensional coding scheme, for distributed matrix multiplication.
- The extra computation at each node allows DEP codes to have a significantly lower recovery threshold than EP codes, leading to a lower communication overhead and decoding complexity.
- Future work: explore more flexible tradeoff between computation and communication/decoding overhead.

Thank you.