Communication Algorithm-Architecture Co-Design for Distributed Deep Learning

Jiayi Huang  Pritam Majumder  Sungkeun Kim  Abdullah Muzahid  Ki Hwan Yum  EJ Kim

UC Santa Barbara (work done at TAMU)  Texas A&M University
Increasing Demand for Distributed Training

- Dataset and model complexity is exploding
Increasing Demand for Distributed Training

- Dataset and model complexity is exploding

Source: Dally, Logarithmic Numbers and Asynchronous Accumulators, The Future of DL Chips
Chips & Compiler Symposium at MLSys'21
Increasing Demand for Distributed Training

- Dataset and model complexity is exploding

Source: Dally, Logarithmic Numbers and Asynchronous Accumulators, The Future of DL Chips
Chips & Compiler Symposium at MLSys’21
Increasing Demand for Distributed Training

- Dataset and model complexity is exploding

GPT-3 – trained on part of an 10,000-GPU cluster* [Brown+ 2020]

*Source: https://developer.nvidia.com/blog/openai-presents-gpt-3-a-175-billion-parameters-language-model/
Data-Parallel Training
Data-Parallel Training

Input Dataset

xPU  xPU  xPU  xPU
Data-Parallel Training

Input Dataset
Data-Parallel Training

Input Dataset

xPU  xPU  xPU  xPU
Data-Parallel Training

Input Dataset

DNN Model

xPU

xPU

xPU

xPU

DNN model redrawn from Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
Data-Parallel Training

DNN model redrawn from Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
Data-Parallel Training

DNN model redrawn from Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
Data-Parallel Training

Input Dataset

DNN Model

Back-Propagation

xPU

DNN model redrawn from Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
Data-Parallel Training

[Diagram showing the process of data-parallel training with an input dataset, DNN model, and an all-reduce operation involving xPUs.]

All-Reduce
(reduce-scatter and all-gather)

DNN model redrawn from Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
Data-Parallel Training

Input Dataset

DNN Model

Weight Update

xPU

xPU

xPU

xPU

DNN model redrawn from Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
Limitations in Existing All-Reduce Algorithms
Limitations in Existing All-Reduce Algorithms

Step 1: 0 1 2 3
Step 2: 1 3 0 2
Step 3: 3 2 1 0

Ring All-Gather:

0 → 1
2 → 3
Limitations in Existing All-Reduce Algorithms

Step 1: 0 1 2 3
Step 2: 1 3 0 2
Step 3: 3 2 1 0

Ring All-Gather:
- Step 1: 0 → 1
- Step 2: 0 → 2
- Step 3: 0 → 3

Linear to #nodes - 1
- Long latency for small data

- 3 steps

- 3 steps

- 3 steps
Limitations in Existing All-Reduce Algorithms

- **Long latency for small data**
- **Applied Well on Various Topologies**

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>(Small data) Latency</th>
<th>Large data</th>
<th>Applied Well on Various Topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring [Patarasuk+Yuan JPDC’09]</td>
<td>high</td>
<td>optimal</td>
<td>✗</td>
</tr>
</tbody>
</table>

**Ring All-Gather**
Linear to #nodes - 1

Step 1: 0 1 2 3
Step 2: 1 3 0 2
Step 3: 3 2 1 0

3 steps
## Limitations in Existing All-Reduce Algorithms

### Algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>(Small data) Latency</th>
<th>Large data Bandwidth</th>
<th>Large data Contention</th>
<th>Applied Well on Various Topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring [Patarasuk+Yuan JPDC’09]</td>
<td>high</td>
<td>optimal</td>
<td>none</td>
<td>✓</td>
</tr>
</tbody>
</table>

### Diagrams

- **Ring All-Gather**
  - Step 1: 0 → 1 → 2 → 3
  - Step 2: 1 → 0, 3 → 2
  - Step 3: 2 → 3 → 0, 1 → 2

- **Double Binary Tree Broadcast (All-Gather)**
  - 0 → 1, 3 → 2

*Linear to #nodes - 1
Long latency for small data*
Limitations in Existing All-Reduce Algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>(Small data)</th>
<th>Large data</th>
<th>Applied Well on Various Topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Latency</td>
<td>Bandwidth</td>
<td>Contention</td>
</tr>
<tr>
<td>Ring [Patarasuk+Yuan JPDC’09]</td>
<td>high</td>
<td>optimal</td>
<td>none</td>
</tr>
<tr>
<td>Double binary tree [Sanders+ JPC’09]</td>
<td>low</td>
<td>optimal</td>
<td>high</td>
</tr>
</tbody>
</table>

Linear to #nodes - 1
Long latency for small data

Double Binary Tree Broadcast (All-Gather)

Ring All-Gather

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Step 2</th>
<th>Step 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Step 1: 0 1 2 3
Step 2: 1 3 0 2
Step 3: 3 2 1 0

Double binary tree broadcast (All-Gather)
Limitations in Existing All-Reduce Algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>(Small data) Latency</th>
<th>Large data Bandwidth</th>
<th>Large data Contention</th>
<th>Applied Well on Various Topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring [Patarasuk+Yuan JPDC’09]</td>
<td>high</td>
<td>optimal</td>
<td>none</td>
<td>✓</td>
</tr>
<tr>
<td>Double binary tree [Sanders+ JPC’09]</td>
<td>low</td>
<td>optimal</td>
<td>high</td>
<td>× (Topology-oblivious)</td>
</tr>
</tbody>
</table>

Topologies:
- **Ring All-Gather**
  - Linear to #nodes - 1
  - Long latency for small data

- **Double Binary Tree Broadcast (All-Gather)**
  - Topology-oblivious: node 1 to 2 crosses node 3 with 2 hops
  - Network contention for large data

Steps:
- **Step 1**
  - Node 0 to 1
- **Step 2**
  - Node 1 to 2
- **Step 3**
  - Node 2 to 3
## Limitations in Existing All-Reduce Algorithms

### Algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>(Small data) Latency</th>
<th>Large data Bandwidth</th>
<th>Large data Contention</th>
<th>Applied Well on Various Topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring [Patarasuk+Yuan JPDC’09]</td>
<td>high</td>
<td>optimal</td>
<td>none</td>
<td>✓</td>
</tr>
<tr>
<td>Double binary tree [Sanders+ JPC’09]</td>
<td>low</td>
<td>optimal</td>
<td>high</td>
<td>× (Topology-oblivious)</td>
</tr>
<tr>
<td>2D-Ring [Ying+ NeurIPsW’18]</td>
<td>low</td>
<td>sub-optimal</td>
<td>none</td>
<td>× (2D Torus/Mesh)</td>
</tr>
</tbody>
</table>

### Diagrams
- **Ring All-Gather**: Linear to #nodes - 1
- **Double Binary Tree Broadcast (All-Gather)**: Log N steps
- **Step 1**: 0 1 2 3
- **Step 2**: 1 3 0 2
- **Step 3**: 3 2 1 0
- **Long latency for small data**: Node 1 to 2 crosses node 3 with 2 hops
- **Network contention for large data**: For large data, contention may occur.
Limitations in Existing All-Reduce Algorithms

### Algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>(Small data)</th>
<th>Large data</th>
<th>Applied Well on Various Topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Latency</td>
<td>Bandwidth</td>
<td>Contention</td>
</tr>
<tr>
<td>Ring ([\text{Patarasuk+Yuan JPDC'09}])</td>
<td>high</td>
<td>optimal</td>
<td>none</td>
</tr>
<tr>
<td>Double binary tree ([\text{Sanders+ JPC'09}])</td>
<td>low</td>
<td>optimal</td>
<td>high</td>
</tr>
<tr>
<td>2D-Ring ([\text{Ying+ NeurIPsW'18}])</td>
<td>low</td>
<td>sub-optimal</td>
<td>none</td>
</tr>
<tr>
<td>HDRM ([\text{Dong+ HPCA'20}])</td>
<td>low</td>
<td>optimal</td>
<td>none</td>
</tr>
</tbody>
</table>

- **Ring**: Optimal latency, high bandwidth, none contention.
- **Double binary tree**: Low latency, optimal bandwidth, high contention.
- **2D-Ring**: Low latency, sub-optimal bandwidth, none contention.
- **HDRM**: Low latency, optimal bandwidth, none contention.
MultiTree: Algorithm-Architecture Co-Design

- Topology-aware All-Reduce Algorithm
  - Low latency and high bandwidth, applicable to different topologies
- Hardware-based All-Reduce Scheduling
  - Contention-free communication
- Message-based Flow Control
  - Exploit bulk transfer of large gradients for near perfect link bandwidth
MultiTree: Algorithm-Architecture Co-Design

- Topology-aware All-Reduce Algorithm
  Low latency and high bandwidth, applicable to different topologies

- Hardware-based All-Reduce Scheduling
  Contention-free communication

- Message-based Flow Control
  Exploit bulk transfer of large gradients for near perfect link bandwidth

- Insight
  - Tree levels closer to leaves are denser than tree levels closer to roots
  - Top-down for tree construction: move more communications to roots
MultiTree: Algorithm-Architecture Co-Design

- Topology-aware All-Reduce Algorithm
  Low latency and high bandwidth, applicable to different topologies

- Hardware-based All-Reduce Scheduling
  Contention-free communication

- Message-based Flow Control
  Exploit bulk transfer of large gradients for near perfect link bandwidth

- Insight
  - Tree levels closer to leaves are denser than tree levels closer to roots
  - Top-down for tree construction: move more communications to roots

- Approach: tree constructions as a link allocation problem
  - Allocate link for each time step (level) to build the trees progressively
MultiTree Construction Example (Time Step 1)

Construct 4 spanning trees for a 4-node system

Link allocation for time step 1 (tree level 1)
MultiTree Construction Example (Time Step 1)

Construct 4 spanning trees for a 4-node system

Link allocation for time step 1 (tree level 1)
MultiTree Construction Example (Time Step 1)

Construct 4 spanning trees for a 4-node system

Link allocation for time step 1 (tree level 1)
MultiTree Construction Example (Time Step 1)

Construct 4 spanning trees for a 4-node system

Link allocation for time step 1 (tree level 1)
MultiTree Construction Example (Time Step 1)

Construct 4 spanning trees for a 4-node system

Link allocation for time step 1 (tree level 1)
MultiTree Construction Example (Time Step 1)

Construct 4 spanning trees for a 4-node system

Link allocation for time step 1 (tree level 1)
MultiTree Construction Example (Time Step 1)

Construct 4 spanning trees for a 4-node system

Run out of links for time step 1

Link allocation for time step 1 (tree level 1)
MultiTree Construction Example (Time Step 2)

Construct 4 spanning trees for a 4-node system

Run out of links for time step 1

Link allocation for time step 2 (tree level 2)
MultiTree Construction Example (Time Step 2)

Construct 4 spanning trees for a 4-node system

Run out of links for time step 1

Link allocation for time step 2 (tree level 2)
MultiTree Construction Example (Time Step 2)

Construct 4 spanning trees for a 4-node system

Run out of links for time step 1

Link allocation for time step 2 (tree level 2)
MultiTree All-Reduce: Reduce-Scatter

Reduce-Scatter (reduction from leaf level to root)

All-Gather (broadcast from root to leaf level)
MultiTree All-Reduce: Reduce-Scatter

Reduce-Scatter (reduction from leaf level to root)

All-Gather (broadcast from root to leaf level)
MultiTree All-Reduce: Reduce-Scatter

Reduce-Scatter (reduction from leaf level to root)

All-Gather (broadcast from root to leaf level)
MultiTree All-Reduce: Reduce-Scatter

Reduce-Scatter (reduction from leaf level to root)

All-Gather (broadcast from root to leaf level)
MultiTree All-Reduce: All-Gather

Reduce-Scatter (reduction from leaf level to root)

All-Gather (broadcast from root to leaf level)

[Diagram showing a tree structure with nodes labeled 0, 1, 2, 3 and arrows indicating the direction of data flow.]
MultiTree All-Reduce: All-Gather

Reduce-Scatter (reduction from leaf level to root)

All-Gather (broadcast from root to leaf level)
MultiTree All-Reduce: All-Gather

Reduce-Scatter (reduction from leaf level to root)

All-Gather (broadcast from root to leaf level)
MultiTree All-Reduce: All-Gather

Reduce-Scatter (reduction from leaf level to root)

All-Gather (broadcast from root to leaf level)
Hardware-based All-Reduce Scheduling and Example

- Message Command (Instruction): stored in an all-reduce schedule table entry

<table>
<thead>
<tr>
<th>Op</th>
<th>FlowID</th>
<th>Parent</th>
<th>Children</th>
<th>Step</th>
<th>Start Addr</th>
<th>Size</th>
</tr>
</thead>
</table>
Hardware-based All-Reduce Scheduling and Example

- **Message Command (Instruction):** stored in an all-reduce schedule table entry

<table>
<thead>
<tr>
<th>Op</th>
<th>FlowID</th>
<th>Parent</th>
<th>Children</th>
<th>Step</th>
<th>Start Addr</th>
<th>Size</th>
</tr>
</thead>
</table>

- **Op:** Reduce, Gather, NOP
**Message Command (Instruction):** stored in an all-reduce schedule table entry

<table>
<thead>
<tr>
<th>Op</th>
<th>FlowID</th>
<th>Parent</th>
<th>Children</th>
<th>Step</th>
<th>Start Addr</th>
<th>Size</th>
</tr>
</thead>
</table>

- **Op:** Reduce, Gather, NOP
- **FlowID:** the ID of the spanning tree root
## Hardware-based All-Reduce Scheduling and Example

- **Message Command (Instruction):** stored in an all-reduce schedule table entry

<table>
<thead>
<tr>
<th>Op</th>
<th>FlowID</th>
<th>Parent</th>
<th>Children</th>
<th>Step</th>
<th>Start Addr</th>
<th>Size</th>
</tr>
</thead>
</table>

- **Op:** Reduce, Gather, NOP
- **FlowID:** the ID of the spanning tree root
- **Parent:** for `Reduce` in tree reduction
- **Children:** for `Gather` in tree broadcast
Hardware-based All-Reduce Scheduling and Example

- **Message Command (Instruction):** stored in an all-reduce schedule table entry

<table>
<thead>
<tr>
<th>Op</th>
<th>FlowID</th>
<th>Parent</th>
<th>Children</th>
<th>Step</th>
<th>Start Addr</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce, Gather, NOP</td>
<td>FlowID: the ID of the spanning tree root</td>
<td>Parent: for Reduce in tree reduction</td>
<td>Children: for Gather in tree broadcast</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Op:** Reduce, Gather, NOP
- **FlowID:** the ID of the spanning tree root
- **Parent:** for Reduce in tree reduction
- **Children:** for Gather in tree broadcast
## Hardware-based All-Reduce Scheduling and Example

- **Message Command (Instruction):** stored in an all-reduce schedule table entry

<table>
<thead>
<tr>
<th>Op</th>
<th>FlowID</th>
<th>Parent</th>
<th>Children</th>
<th>Step</th>
<th>Start Addr</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce</td>
<td>3</td>
<td>1</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
</tr>
<tr>
<td>Reduce</td>
<td>1</td>
<td>1</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
</tr>
<tr>
<td>Reduce</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
</tr>
<tr>
<td>Gather</td>
<td>0</td>
<td>nil</td>
<td>1</td>
<td>2</td>
<td>nil</td>
<td>nil</td>
</tr>
<tr>
<td>Gather</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
</tr>
</tbody>
</table>

- **Op:** Reduce, Gather, NOP
- **FlowID:** the ID of the spanning tree root
- **Parent:** for Reduce in tree reduction
- **Children:** for Gather in tree broadcast

### Accelerator 0

- **Reduce-Scatter (Reduction):**
  - Step 1: Node 3
  - Step 2: Node 2, Node 1, Node 3, Node 0

- **All-Gather (Broadcast):**
  - Step 3: Node 0, Node 1, Node 2, Node 3
  - Step 4: Node 2, Node 1, Node 3, Node 0
Hardware-based All-Reduce Scheduling and Example

- **Message Command (Instruction):** stored in an all-reduce schedule table entry
  - **Op:** Reduce, Gather, NOP
  - **FlowID:** the ID of the spanning tree root
  - **Parent:** for Reduce in tree reduction
  - **Children:** for Gather in tree broadcast

### Accelerator 0

<table>
<thead>
<tr>
<th>Op</th>
<th>FlowID</th>
<th>Parent</th>
<th>Children</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce</td>
<td>3</td>
<td>1</td>
<td>nil, nil</td>
<td>nil</td>
</tr>
<tr>
<td>Reduce</td>
<td>1</td>
<td>1</td>
<td>nil, nil</td>
<td>nil</td>
</tr>
<tr>
<td>Reduce</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>nil</td>
</tr>
<tr>
<td>Gather</td>
<td>0</td>
<td>nil</td>
<td>1, 2</td>
<td>nil</td>
</tr>
<tr>
<td>Gather</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>nil</td>
</tr>
</tbody>
</table>

- **Reduce-Scatter (Reduction):**
  - Step 1: 3
  - Step 2: 2
  - Step 3: 0
  - Step 4: 2

- **All-Gather (Broadcast):**
  - Step 1: 0
  - Step 2: 1
  - Step 3: 2
  - Step 4: 3
Evaluation – Bandwidth (top) and DNN Training Time (bottom)
Evaluation — Bandwidth (top) and DNN Training Time (bottom)
Evaluation — Bandwidth (top) and DNN Training Time (bottom)
Evaluation – Bandwidth (top) and DNN Training Time (bottom)

- **BW in Torus and BiGraph**
  - MultiTree achieves low latency and high BW
Evaluation – Bandwidth (top) and DNN Training Time (bottom)

- BW in Torus and BiGraph
  - MultiTree achieves low latency and high BW
  - In Torus, 2D-Ring > Ring > DBTree
Evaluation — Bandwidth (top) and DNN Training Time (bottom)

- **BW in Torus and BiGraph**
  - MultiTree achieves low latency and high BW
  - In Torus, 2D-Ring > Ring > DBTree
  - In Bigraph, HDRM good at latency and BW, but worse than MultiTree
Evaluation – Bandwidth (top) and DNN Training Time (bottom)

- **BW in Torus and BiGraph**
  - MultiTree achieves low latency and high BW
  - In Torus, 2D-Ring > Ring > DBTree
  - In Bigraph, HDRM good at latency and BW, but worse than MultiTree
  - Ring has good BW while DBTree has good latency in BiGraph
Evaluation — Bandwidth (top) and DNN Training Time (bottom)

- BW in Torus and BiGraph
  - MultiTree achieves low latency and high BW
  - In Torus, 2D-Ring > Ring > DBTree
  - In Bigraph, HDRM good at latency and BW, but worse than MultiTree
  - Ring has good BW while DBTree has good latency in BiGraph

- DNN Training Time in 8x8 Torus
Evaluation – Bandwidth (top) and DNN Training Time (bottom)

- **BW in Torus and BiGraph**
  - MultiTree achieves low latency and high BW
  - In Torus, 2D-Ring > Ring > DBTree
  - In Bigraph, HDRM good at latency and BW, but worse than MultiTree
  - Ring has good BW while DBTree has good latency in BiGraph

- **DNN Training Time in 8x8 Torus**
  - 2.3x and 1.56x *communication* speedup over Ring and 2D-Ring
  - Up to 81% and 31% training time reduction compared to Ring and 2D-Ring
Evaluation — Bandwidth (top) and DNN Training Time (bottom)

-BW in Torus and BiGraph
  - MultiTree achieves low latency and high BW
  - In Torus, 2D-Ring > Ring > DBTree

-Bigraph
  - HDRM good at latency and BW, but worse than MultiTree
  - 2D-Ring has good BW while DBTree has good latency in BiGraph

DNN Training Time in 8x8 Torus
  - 2.3x and 1.56x communication speedup over Ring and 2D-Ring
  - Up to 81% and 31% training time reduction compared to Ring and 2D-Ring

More in the paper

- Hardware-based scheduling control and datapath design
- Message-based flow control
- More evaluation settings and results
  - Mesh and Fat-Tree network topologies with different scales
  - Scalability study
  - Communication and computation overlap for DNN Training

Hardware-based scheduling control and datapath design
Message-based flow control
More evaluation settings and results
- Mesh and Fat-Tree network topologies with different scales
- Scalability study
- Communication and computation overlap for DNN Training

Up to 81% and 31% training time reduction compared to Ring and 2D-Ring

More in the paper
Communication Algorithm-Architecture Co-Design for Distributed Deep Learning

Jiayi Huang   Pritam Majumder   Sungkeun Kim
Abdullah Muzahid   Ki Hwan Yum   EJ Kim

UC Santa Barbara (work done at TAMU)  Texas A&M University