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 Deep Learning

- Dataset and model sizes are big and increasing

*Source: Data Age 2025, sponsored by Seagate with data from IDC Global DataSphere, Nov 2018*
Increasing Demand for Distributed Deep Learning

- Dataset and model sizes are big and increasing

- Train ImageNet in 1 hour – 256 GPUs [Goyal+ 2017]

- AlphaZero – 5000 TPU v1 for games, 64 TPU v2 for training [Silver+ Science’18]
Data-Parallel Training – All-Reduce

Figure source: Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
Data-Parallel Training – All-Reduce

Figure source: Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
Data-Parallel Training – All-Reduce

Figure source: Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
Data-Parallel Training – All-Reduce

Figure source: Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
## Data-Parallel Training – All-Reduce

![Diagram of All-Reduce](image-source)

---

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>(Small data) Latency</th>
<th>(Large data) Bandwidth-optimal</th>
<th>(Large data) Contention-free</th>
<th>Applied Well on Various Topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring [Patarasuk+Yuan JPDC’09]</td>
<td>high</td>
<td>✓</td>
<td>✓</td>
<td>✓ (Topology-oblivious)</td>
</tr>
<tr>
<td>Double binary tree [Sanders+ JPC’09]</td>
<td>low</td>
<td>✓</td>
<td>✗</td>
<td>✗ (2D Torus/Mesh)</td>
</tr>
<tr>
<td>2D-Ring [Ying+ NeurIPS’18]</td>
<td>low</td>
<td>✗</td>
<td>✓</td>
<td>✗ (BiGraph)</td>
</tr>
<tr>
<td>HDRM [Dong+ HPCA’20]</td>
<td>low</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Figure source: Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019
# Data-Parallel Training – All-Reduce

## MultiTree: Algorithm-Architecture Co-Design
- Topology-aware All-Reduce
- Hardware All-Reduce Scheduling
- Big Message Flow Control

**Figure source:** Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>(Small data) Latency</th>
<th>(Large data) Bandwidth-optimal</th>
<th>(Large data) Contention-free</th>
<th>Applied Well on Various Topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring [Patarasuk+Yuan JPDC’09]</td>
<td>high</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Double binary tree [Sanders+ JPC’09]</td>
<td>low</td>
<td>✓</td>
<td>✗</td>
<td>✗ (Topology-oblivious)</td>
</tr>
<tr>
<td>2D-Ring [Ying+ NeurIPS’18]</td>
<td>low</td>
<td>✗</td>
<td>✓</td>
<td>✗ (2D Torus/Mesh)</td>
</tr>
<tr>
<td>HDRM [Dong+ HPCA’20]</td>
<td>low</td>
<td>✓</td>
<td>✓</td>
<td>✗ (BiGraph)</td>
</tr>
</tbody>
</table>
# Data-Parallel Training – All-Reduce

![All-Reduce Diagram](image)

**Figure source:** Ben-Nun+ ACM Computing Surveys, vol. 52, no. 4, August 2019

---

## MultiTree: Algorithm-Architecture Co-Design
- Topology-aware All-Reduce
- Hardware All-Reduce Scheduling
- Big Message Flow Control

---

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>(Small data) Latency</th>
<th>(Large data) Bandwidth-optimal</th>
<th>(Large data) Contention-free</th>
<th>Applied Well on Various Topologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring [Patarasuk+Yuan JPDC’09]</td>
<td>high</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Double binary tree [Sanders+ JPC’09]</td>
<td>low</td>
<td>✓</td>
<td>✓</td>
<td>× (Topology-oblivious)</td>
</tr>
<tr>
<td>2D-Ring [Ying+ NeurIPSW’18]</td>
<td>low</td>
<td>✓</td>
<td>✓</td>
<td>× (2D Torus/Mesh)</td>
</tr>
<tr>
<td>HDRM [Dong+ HPCA’20]</td>
<td>low</td>
<td>✓</td>
<td>✓</td>
<td>× (BiGraph)</td>
</tr>
<tr>
<td><strong>MultiTree (Ours)</strong></td>
<td>low</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Topology-aware MultiTree All-Reduce
Topology-aware MultiTree All-Reduce

- Topology-aware spanning trees instead of rings
- Combine tree constructions with message scheduling — Contention-free
Topology-aware MultiTree All-Reduce

- Topology-aware spanning trees instead of rings
- Combine tree constructions with message scheduling – Contention-free

Insight: tree levels closer to leaves are denser than tree levels closer to roots
Topology-aware MultiTree All-Reduce

- Topology-aware spanning trees instead of rings
- Combine tree constructions with message scheduling – Contention-free

- Insight: *tree levels closer to leaves are denser than tree levels closer to roots*
- Top-down approach – move more communications closer to roots
Topology-aware MultiTree All-Reduce

- Topology-aware spanning trees instead of rings
- Combine tree constructions with message scheduling – Contention-free

- Insight: tree levels closer to leaves are denser than tree levels closer to roots
- Top-down approach – move more communications closer to roots

- Constructing trees – link allocation problem (global coordination)
  - Allocate link for each step (level) to build the trees progressively
Multitree Example (Time Step 1)

X+, Y+ Links

X–, Y– Links
Multitree Example (Time Step 1)

X+, Y+ Links

X–, Y– Links
Multitree Example (Time Step 1)

X+, Y+ Links

X-, Y- Links
Multitree Example (Time Step 1)

X+, Y+ Links

0  1  2
3  4  5
6  7  8

X−, Y− Links

0  1  2
3  4  5
6  7  8
Multitree Example (Time Step 2)

X+, Y+ Links

X−, Y− Links
Multitree Example (Time Step 2)

X+, Y+ Links

X–, Y– Links
Multitree Example (Time Step 2)

X+, Y+ Links

X–, Y– Links
Multitree Example (Time Step 2)

X+, Y+ Links

X–, Y– Links
Multitree Example (Time Step 3)

X+, Y+ Links

X−, Y− Links
Multitree Example (Time Step 3)

X+, Y+ Links

X-, Y- Links
Multitree Reduce-Scatter Phase

X-, Y- Links

X+, Y+ Links
Multitree Reduce-Scatter Phase

X−, Y− Links

X+, Y+ Links
Multitree Reduce-Scatter Phase

X-, Y- Links

X+, Y+ Links
Multitree Reduce-Scatter Phase

X-, Y- Links

X+, Y+ Links
Multitree All-Gather Phase

X+, Y+ Links

X–, Y– Links
Multitree All-Gather Phase

X+, Y+ Links

X-, Y- Links
Multitree All-Gather Phase

X+, Y+ Links

X–, Y– Links
Multitree All-Gather Phase

X+, Y+ Links

X–, Y– Links
Hardware-based All-Reduce Scheduling

- **Message Command (Instruction)**

<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>

- Stored in an all-reduce schedule table entry
- **Op:** Reduce, Gather, NOP
- **FlowID:** the ID of the reduction/broadcast tree
## Hardware-based All-Reduce Scheduling

- **Message Command (Instruction)**
  - Stored in an all-reduce schedule table entry
  - **Op**: Reduce, Gather, NOP
  - **FlowID**: the ID of the reduction/broadcast tree

### 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>8</td>
<td>2</td>
<td>nil nil nil nil nil</td>
<td>0</td>
</tr>
<tr>
<td>Reduce</td>
<td>4</td>
<td>1</td>
<td>nil nil nil nil nil</td>
<td>1</td>
</tr>
<tr>
<td>Reduce</td>
<td>5</td>
<td>2</td>
<td>nil nil nil nil nil</td>
<td>1</td>
</tr>
<tr>
<td>Reduce</td>
<td>7</td>
<td>1</td>
<td>2 nil nil nil nil</td>
<td>1</td>
</tr>
<tr>
<td>Reduce</td>
<td>1</td>
<td>3</td>
<td>nil nil nil nil</td>
<td>2</td>
</tr>
<tr>
<td>Reduce</td>
<td>2</td>
<td>2</td>
<td>nil nil nil nil</td>
<td>2</td>
</tr>
<tr>
<td>Reduce</td>
<td>3</td>
<td>3</td>
<td>1 2 nil nil</td>
<td>2</td>
</tr>
<tr>
<td>Reduce</td>
<td>6</td>
<td>6</td>
<td>1 nil nil nil</td>
<td>2</td>
</tr>
<tr>
<td>Reduce</td>
<td>0</td>
<td>nil</td>
<td>1 2 3 6 nil</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Op</th>
<th>FlowID</th>
<th>Parent</th>
<th>Children</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gather</td>
<td>0</td>
<td>nil</td>
<td>1 2 3 6 nil</td>
<td>3</td>
</tr>
<tr>
<td>Gather</td>
<td>1</td>
<td>1</td>
<td>3 nil nil nil</td>
<td>4</td>
</tr>
<tr>
<td>Gather</td>
<td>2</td>
<td>2</td>
<td>nil nil nil nil</td>
<td>4</td>
</tr>
<tr>
<td>Gather</td>
<td>4</td>
<td>1</td>
<td>nil nil nil nil</td>
<td>5</td>
</tr>
<tr>
<td>Gather</td>
<td>5</td>
<td>2</td>
<td>nil nil nil nil</td>
<td>5</td>
</tr>
<tr>
<td>Gather</td>
<td>6</td>
<td>6</td>
<td>1 nil nil nil</td>
<td>5</td>
</tr>
<tr>
<td>Gather</td>
<td>7</td>
<td>6</td>
<td>nil nil nil nil</td>
<td>5</td>
</tr>
<tr>
<td>Gather</td>
<td>8</td>
<td>2</td>
<td>nil nil nil nil</td>
<td>5</td>
</tr>
</tbody>
</table>
All-Reduce Schedule Control and Datapath

### Receiving Messages

- Reduce Message from Network
- Gather Message from Network

### Sending Messages

- Reduce/Gather/NOP

### Timings

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $\text{Timestep Counter} = \text{null}$
- $\text{Counter} = \text{null}$

### Control Logic

1. **Reduce/Gather/NOP**
2. **Lockstep Down Counter**
3. **Increment Counter**
4. **Reduction Logic**
5. **Reduction Logic**
6. **Gather/Reduce Logic**

### Major Events

- **Idle**
- **DMA Request**
- **DMA Response**
- **Data**

### Side Events

- **Reduce/Gather Message to Network**
- **Receive Messages**
- **Send Messages**

### Conditions

- **Step > Timestep**
- **FlowID, Start Addr, Size**
- **Reduce/Gather**
All-Reduce Schedule Control and Datapath

<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>

- Reduce Message from Network
- Gather Message from Network
- Reduce/Gather/NOP

1. Step > Timestep
2. Increment Counter
3. Reduce/Gather Message to Network
4. Reduction Logic
5. [Reduce] Send to Parent (FlowID, Start Addr, Size)
6. [Gather] Send to Children (FlowID, Start Addr, Size)
All-Reduce Schedule Control and Datapath

### Receiving Messages
- Reduce Message from Network
- Gather Message from Network

### Sending Messages
- Reduce/Gather/NOP
- [Gather] Send to Children (FlowID, Start Addr, Size)
- [Reduce] Send to Parent (FlowID, Start Addr, Size)

---

### Table

<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>

---

### Diagram

1. [Gather] Send to Children
2. [Reduce] Send to Parent
3. Increment Counter
4. Reduction Logic
5. [Reduce] Send to Parent
6. [Gather] Send to Children

---

All-Reduce Schedule Control and Datapath

---

11/16
All-Reduce Schedule Control and Datapath

### Receiving Messages
- **Reduce Message from Network**
- **Gather Message from Network**

### Sending Messages
- **Reduce/Gather/NOP**
- **[NOP] Set Counter**
- **Lockstep Down Counter**
- **Decode**
- **[Gather] Send to Children (FlowID, Start Addr, Size)**
- **[Reduce] Send to Parent (FlowID, Start Addr, Size)**

### Reduction Logic
- **Timestep Counter**
- **Increment Counter**
- **Step > Timestep**

### Table

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Steps:**
1. Reduce/Gather/NOP
2. [NOP] Set Counter
3. Lockstep Down Counter
4. Reduction Logic
5. [Gather] Send to Children (FlowID, Start Addr, Size)
6. [Reduce] Send to Parent (FlowID, Start Addr, Size)
All-Reduce Schedule Control and Datapath
Big Message Flow Control

- Big gradient exchange has high head flit overhead (6%)
Big Message Flow Control

- Big gradient exchange has high head flit overhead (6%)

- Message-based flow control

```
<table>
<thead>
<tr>
<th>Head Sub-MESSAGE</th>
<th>Body Sub-MESSAGE</th>
<th>...</th>
<th>Tail Sub-MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head Sub-PACKET</td>
<td>Body Sub-PACKET</td>
<td>...</td>
<td>Body Sub-PACKET</td>
</tr>
<tr>
<td>Head Flit</td>
<td>Body Flit</td>
<td>...</td>
<td>Sub Tail Flit</td>
</tr>
<tr>
<td>n Flits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Body Sub-PACKET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Body Flit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sub Tail Flit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>n-1 Flits</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Head Sub-MESSAGE</th>
<th>Body Sub-MESSAGE</th>
<th>...</th>
<th>Tail Sub-MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head Sub-PACKET</td>
<td>Body Sub-PACKET</td>
<td>...</td>
<td>Body Sub-PACKET</td>
</tr>
<tr>
<td>Head Flit</td>
<td>Body Flit</td>
<td>...</td>
<td>Sub Tail Flit</td>
</tr>
<tr>
<td>n Flits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Body Sub-PACKET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Body Flit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sub Tail Flit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>n-1 Flits</td>
</tr>
</tbody>
</table>
```

Big gradient exchange has high head flit overhead (6%)

- Message-based flow control

Flit: flow control digit
Results – Bandwidth on Directed Networks

Results for different network topologies (MultiTree, 2D-Ring, 2BTree, Ring) under various data sizes (64KiB, 512KiB, 4MiB, 32MiB) in 4×4 Torus and Mesh networks.
Double binary tree (2BTree) is very unfriendly to Torus and Mesh
Results – Bandwidth on Directed Networks

- Double binary tree (2BTree) is very unfriendly to Torus and Mesh
- Ring faces sever link under utilizations
Double binary tree (2BTree) is very unfriendly to Torus and Mesh
Ring faces sever link under utilizations
2D-Ring bandwidth sub-optimal, sending more data (could be twice amount)
Double binary tree (2BTree) is very unfriendly to Torus and Mesh
Ring faces sever link under utilizations
2D-Ring bandwidth sub-optimal, sending more data (could be twice amount)
MultiTree solves all the above problems
Results – Bandwidth on Switch-based Networks

![Graph 1: Bandwidth vs. Data Size on 16-Node Fat-Tree](image1)

- **MultiTree**
- **2BTree**
- **Ring**

![Graph 2: Bandwidth vs. Data Size on 32-Node BiGraph](image2)

- **MultiTree**
- **HDRM**
- **2BTree**
- **Ring**

All-Reduce Data Size on 16-Node Fat-Tree

All-Reduce Data Size on 32-Node 4 x 8 BiGraph
Ring achieves good bandwidth for large data (long latency for small data)
Ring achieves good bandwidth for large data (long latency for small data)

Double binary tree has good latency for small data, but bad at bandwidth
Results – Bandwidth on Switch-based Networks

- Ring achieves good bandwidth for large data (long latency for small data)
- Double binary tree has good latency for small data, but bad at bandwidth
- MultiTree works well for both small and large data
Results – DNN Benchmarks on 8x8 Torus
Results – DNN Benchmarks on 8x8 Torus

- MultiTree > 2D-Ring > Ring > 2BTree (Double binary tree)
Summary

- Identifying inefficiencies in existing all-reduce algorithms

- MultiTree All-Reduce: algorithm-architecture co-design
  - Topology-aware and link usage coordination
  - Hardware-based all-reduce scheduling
  - Big message flow control for big gradients

- Achieves low latency as well as high throughput
  - Beats prior work with 2.5x improvement compared to Ring all-reduce