ASTRA-sim2.0: Modeling Hierarchical Networks and Disaggregated Systems for Large-model Training at Scale

William Won¹*  Taekyung Heo¹*  Saeed Rashidi¹*  Srinivas Sridharan²  Sudarshan Srinivasan³  Tushar Krishna¹

¹Georgia Institute of Technology  ²Meta  ³Intel  *Equal contribution

Joint work with Georgia Tech, Meta, and Intel

https://astra-sim.github.io

https://synergy.ece.gatech.edu
Outline

• Distributed Training
• Background
• ASTRA-sim
• Limitations of ASTRA-sim
• ASTRA-sim2.0
• Case Studies and Results
• Conclusion
Large Language Models (LLMs)
Models are Becoming Larger

• Deep learning (DL) **models are scaling** at an unprecedented rate

https://huggingface.co/blog/large-language-models
Dataset is also Becoming Larger

"570 GB of uncompressed text data"

"1.56 trillion words"

As a language model, I am a pre-trained model that has been trained on a large corpus of text data to learn patterns and relationships between words, phrases, and sentences. The specific dataset used to train me depends on the version of the GPT architecture that I am built on top of.

For example, the largest version of the GPT architecture, GPT-3, was trained on a diverse and massive dataset of web pages, books, and other sources of text data, totaling over 570 GB of uncompressed text data. This dataset included a wide range of sources, including books, websites, and even programming code, to help provide a broad and comprehensive understanding of the English language.

My training dataset was 1.56 trillion words of text and code. This dataset was collected from a variety of sources, including books, articles, code, and human conversations. The dataset was carefully curated to ensure that it was high-quality and representative of the real world.
Training is a Key Challenge

• Trillion-parameter Models
  • Zeta-scale floating-point operations
  • 10s of TB of memory

• Impractical to be trained on a single GPU
  • 355 GPU-years to train (using NVIDIA V100)
Distributed Training is Inevitable

- Shard model/data across NPUs (Neural Processing Unit)

"thousands of GPUs"

"around 1,000 TPUs"
Distributed Training is Inevitable

- Shard model/data across NPUs (Neural Processing Unit)

Distributed training is necessitated

微量 computer cluster consisting of thousands of GPUs and hundreds of CPUs provided by Microsoft Azure. This massive infrastructure enabled efficient and scalable training of the GPT-3 model, which has 175 billion parameters.

"thousands of GPUs"

"around 1,000 TPUs"
Outline

• Distributed Training
• Background
• ASTRA-sim
• Limitations of ASTRA-sim
• ASTRA-sim2.0
• Case Studies and Results
• Conclusion
Parallelization Strategy

• Model Parallelism (MP)

• Data Parallelism (DP)
Design-space of Distributed Training

- **Workload Parallelization Strategy**
  - Data, Model, Platform Agnostic Hybrid, Platform-aware Hybrid, Pipelined Parallelism
- **Communication Policy and Pattern**
  - Distributed worker, Parameter Server
- **Framework-level Scheduling**
  - Sync/Async, Blocking/Non-blocking
- **Communication Mechanism**
  - Topology-aware Collectives, Send/Recv, RPC
- **Communication Scheduling**
  - Dataflow, Microarchitecture, Flexibility, Sparsity Support
- **Messaging/Transport Layer**
  - LIFO, FIFO, Fusion
- **Compute and Memory Design**
  - TCP, RDMA (+ GPUDirect RDMA)
- **Endpoint Node Design and Connectivity**
  - # links, BW per link, architecture (chip/package/board), NIC offload, compression
- **Fabric Design and Topology**
  - Flat vs. Hierarchical, 2D/3D/4D Torus (TPU), Switch (DGX2), Fully-Connected, Hyper-Cube Mesh
- **Network Implementation**
  - Buffering, Flow-control, Arbitration, Congestion Mgmt

*Figure Courtesy: Srinivas Sridharan (Meta)*
Design-space of Distributed Training

- **DNN Models**: DLRM, ResNet-50, Transformer, GNMT
- **Workload Parallelization Strategy**: Data, Model, Platform Agnostic Hybrid, Platform-aware Hybrid, Pipelined Parallelism

Design-space of distributed training is large and complex

- **Messaging/Transport Layer**: LIFO, FIFO, Fusion, TCP, RDMA (+ GPUDirect RDMA)
- **Communication Mechanism**: Topology-aware Collectives, Send/Recv, RPC, Sync/Async, Blocking/Non-blocking
- **Network Layer**: # links, BW per link, architecture (chip/package/board), NIC offload, compression
- **Fabric Design and Topology**: Flat vs. Hierarchical, 2D/3D/4D Torus (TPU), Switch (DGX2), Fully-Connected, Hyper-Cube Mesh
- **Network Implementation**: Buffering, Flow-control, Arbitration, Congestion Mgmt

*Figure Courtesy: Srinivas Sridharan (Meta)*
Outline

• Distributed Training
• Background
• ASTRA-sim
  • Limitations of ASTRA-sim
  • ASTRA-sim2.0
• Case Studies and Results
• Conclusion
ASTRA-sim

- Supports Data-Parallel, Model-Parallel, Hybrid-Parallel training loops
- Extensible to more training loops

- Ring based, Tree-based, All-to-All based, and multi-phase collectives
- Variety of scheduling policies
- Compute times fed via offline system measurements or compute simulator

- Various topologies, flow-control, link bandwidth, congestion control
- Plug-and-play options
  - Garnet (credit-based)

http://github.com/astra-sim/astra-sim


ASTRA-sim Capabilities

Flat100G

Hier

HierOpt

100 Gb/s

800 Gb/s

NPU
Intra-package scale-up
Inter-package scale-up
Package

Torus3D

Node
Inter-node scale-out communication
Switch

Super-node
Inter-super-node scale-out communication
Node
Switch

Intra-super-node scale-up communication

100 Gb/s
ASTRA-sim Capabilities

ASTRA-sim captures/simulates complex design-space of distributed training

NPU  Intra-package scale-up  Inter-package scale-up  Package

Torus3D

100 Gb/s

100 Gb/s

800 Gb/s

Super-node
Intra-super-node scale-up communication

Node  Switch

Hier

Inter-super-node scale-out communication

Super-node
Intra-super-node scale-up communication

Node  Switch

HierOpt
Outline

• Distributed Training
• Background
• ASTRA-sim
• Limitations of ASTRA-sim
• ASTRA-sim2.0
• Case Studies and Results
• Conclusion
Emerging Platforms

Pipeline Parallelism

Multi-dimensional Networks

Novel Memory Systems through CXL
Limitations of ASTRA-sim

• Rigid parallelization strategy
• Pre-defined network topology with limited scale
• Lack of memory system modeling
Limitations of ASTRA-sim

- Rigid parallelization strategy
- Pre-defined network topology with limited scale

ASTRA-sim cannot model emerging training platforms
Outline

• Distributed Training
• Background
• ASTRA-sim
• Limitations of ASTRA-sim
• ASTRA-sim2.0
• Case Studies and Results
• Conclusion
Overview: ASTRA-sim2.0

- Simulates multi-dimensional networks at scale
- Multi-dimensional topology representation
- Analytical network backend

(a) SW/HW co-design stack of distributed training
(b) Graph-based Execution Engine
(c) Enhanced ASTRA-sim simulation infrastructure

- Supports arbitrary parallelization strategies
- Graph-based Execution Engine
- Execution Traces (ETs)
- Captures memory systems through MemoryAPI
- Local memory model
- Remote memory model
Graph-based Execution Engine

(a) SW/HW co-design stack of distributed training

(b) Graph-based Execution Engine

(c) Enhanced ASTRA-sim simulation infrastructure

(d) Target distributed training infrastructure
Graph-based Execution Engine

- Parallelization is represented in Execution Trace (ET)

```
Memory
Size: 500 MB

Compute
Input: 50 x 10
Weight: 10 x 20

Communication peer-to-peer
100 MB

Compute
Input: 50 x 10
Weight: 10 x 20

Memory
Size: 200 MB

Communication
All-Reduce
300 MB
```
Collecting Execution Trace

- ETs could be easily **collected from PyTorch models**

```python
et = ExecutionGraphObserver()
et.register_callback("et_file.json")
et.start()

# run PyTorch model

et.stop()
et.unregister_callback()
```

Start ET collection

Run model

Stop collection
Multi-dimensional Network Modeling

- **Execution Trace**
  - Parser
  - ML Frameworks
  - Workload Parallelization Strategy
  - Communication Policy and Pattern
  - Framework-level Scheduling
  - Communication Mechanism
  - Communication Scheduling
  - Memory Design
  - Messaging/Transport Layer
  - Endpoint Design and Connectivity
  - Hierarchical Fabric Design and Topology
  - Network Implementation

- **Execution Trace**
  - Memory
  - Compute
  - Comm.
  - Memory
  - Compute
  - Comm.
  - Memory
  - Compute
  - Comm.

- **Network API**
  - Multi-dimensional NPU Fabric
  - NPU
  - NPU
  - NPU

- **Disaggregated Memory Pool**
  - Memory API
  - Remote Memory
  - Remote Memory
  - Remote Memory

- **(a) SW/HW co-design stack of distributed training**
- **(b) Graph-based Execution Engine**
- **(c) Enhanced ASTRA-sim simulation infrastructure**
- **(d) Target distributed training infrastructure**
Network Building Blocks

- Basic building blocks of multi-dimensional networks

  - Ring
  - FullyConnected
  - Switch

- No network congestion while running collective communication

<table>
<thead>
<tr>
<th>Topology Building Block</th>
<th>Topology-aware Collective Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring</td>
<td>Ring</td>
</tr>
<tr>
<td>FullyConnected</td>
<td>Direct</td>
</tr>
<tr>
<td>Switch</td>
<td>HalvingDoubling</td>
</tr>
</tbody>
</table>
Representing Real Systems

• Captures state-of-the-art training platforms

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Component (Networking)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim 1</td>
<td>Chiplet (on-chip)</td>
</tr>
<tr>
<td>Dim 2</td>
<td>Package (NVLink)</td>
</tr>
<tr>
<td>Dim 3</td>
<td>Node (NVLink)</td>
</tr>
<tr>
<td>Dim 4</td>
<td>Pod (NIC)</td>
</tr>
</tbody>
</table>

- **SW(3)_SW(2)**
  - NVIDIA DGX-2
  - NVIDIA DGX-A100

- **R(4)_R(2)_R(2)**
  - Google TPUv4

- **R(4)_SW(2)**
  - Meta Zion
  - NVIDIA DGX-1

- **FC(4)_SW(2)**
  - Intel Habana
Analytical Backend

• Boost up simulation by **analytically modeling communications**

\[
send(src \rightarrow dest, \text{msg\_size}) = \text{#hops}(src \rightarrow dest) \times \text{link\_latency} + \frac{\text{msg\_size}}{\text{link\_bandwidth}}
\]

• Suitable when there's no network contention
  • Topology-aware collective communication
Modeling Emerging Memory Systems

(a) SW/HW co-design stack of distributed training
(b) Graph-based Execution Engine
(c) Enhanced ASTRA-sim simulation infrastructure
(d) Target distributed training infrastructure
Modeling Emerging Memory Systems

• ASTRA-sim2.0 adds a MemoryAPI
  • Could be used for both local/remote memory models

• Local Memory Model
  \[ \text{access}(\text{tensor}\_\text{size}) = \text{memory access latency} + \frac{\text{tensor}\_\text{size}}{\text{memory bandwidth}} \]

• Remote Memory Model
  • Mix and match per design choices (e.g., pipelining multiple stages)

• In-switch Collective Communication
  • Reduction happens on-the-fly inside network switches
Modeling Emerging Memory Systems

• ASTRA-sim2.0 adds a MemoryAPI
  • Could be used for both local/remote memory models

ASTRA-sim2.0 models futuristic training characteristics

• In-switch Collective Communication
  • Reduction happens on-the-fly inside network switches
Outline

• Distributed Training
• Background
• ASTRA-sim
• Limitations of ASTRA-sim
• ASTRA-sim2.0
• Case Studies and Results
• Conclusion
Case Study 1: Conventional vs. Wafer-scale

- Conventional Systems: multi-dimensional with diminishing BW
- Wafer-scale Systems: 1-2D topology with very-high-BW

Case Study 1: Conventional vs. Wafer-scale

• Wafer-scale: 1-2D
  • With very high BW per each Dim

• Conventional Systems: 3-4D
  • With diminishing network BW with higher network dimension

<table>
<thead>
<tr>
<th>Topology</th>
<th>Shape</th>
<th>NPU Size</th>
<th>BW (GB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>W-1D</td>
<td>Switch</td>
<td>512</td>
<td>350, 500, 600</td>
</tr>
<tr>
<td>W-2D</td>
<td>Switch_Switch</td>
<td>32×16</td>
<td>250, 250</td>
</tr>
<tr>
<td>Conv-3D</td>
<td>Ring_FC_Switch</td>
<td>16×8×4</td>
<td>200_100_50</td>
</tr>
<tr>
<td>Conv-4D</td>
<td>Ring_FC_Ring_Switch</td>
<td>2×8×8×4</td>
<td>250_200_100_50</td>
</tr>
</tbody>
</table>
Case Study 1: Result

- **Overhead running multi-dimensional collective** communication
  - W-1D (with higher BW) yields overall best performance
- Conv-4D is still powerful
  - Driving **higher BW per NPU**
Case Study 2: Chunk Scheduling Policy

• **Themis**: Greedy-based chunk scheduling policy
  • To maximize BW utilization of multi-dimensional collective communication

![Diagram of baseline and Themis scheduling]

Case Study 2: Result

- No difference in W-1D, but huge gain in W-2D, Conv-3/4D
- If equal BW/NPU is provisioned, yields near identical performance
  - Regardless of network dimensionality

<table>
<thead>
<tr>
<th>Network Dimension</th>
<th>All-Reduce (1GB)</th>
<th>DLRM</th>
<th>GPT-3</th>
<th>T-1T</th>
</tr>
</thead>
<tbody>
<tr>
<td>W-1D-350</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W-1D-500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W-1D-600</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W-2D-500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv-3D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv-4D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Greedy Collective Scheduler (Themis)

- Compute
- Exposed Comm.
Case Study 3: Comparing Memory Systems

• ZeRO-Infinity: leveraging local memory (NVMe)
• HierMem: disaggregated memory systems with in-switch collective
Case Study 3: Comparing Memory Systems

• ZeRO-Infinity: Baseline

• HierMem:
  • Baseline: equivalent configuration as ZeRO-Infinity
  • Opt: fine-tuned configuration for Mixture-of-Experts (MoE) Model

<table>
<thead>
<tr>
<th></th>
<th>ZeRO-Infinity</th>
<th>HierMem (Baseline)</th>
<th>HierMem (Opt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU Peak Perf (TFLOPS)</td>
<td>2048</td>
<td>2048</td>
<td>2048</td>
</tr>
<tr>
<td>GPU Local HBM BW (GB/sec)</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
</tr>
<tr>
<td>In-node Pooled Fabric BW (GB/sec)</td>
<td>-</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>Num of Out-node Switches</td>
<td>-</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Num of Remote Memory Groups</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>Remote Mem Group BW (GB/sec)</td>
<td>100</td>
<td>100</td>
<td>500</td>
</tr>
</tbody>
</table>
Case Study 3: Result

• ZeRO-Infinity and HierMem (baseline) is near-identical
• Fine-tuned HierMem shows 4.6x better runtime
  • In-switch collective communication reduces exposed communication
Case Study 3: Result

- ZeRO-Infinity and HierMem (baseline) is near-identical
- Fine-tuned HierMem shows 4.6x better runtime

ASTRA-sim2.0 enables design-space exploration of emerging training platforms
Outline

• Distributed Training
• Background
• ASTRA-sim
• Limitations of ASTRA-sim
• ASTRA-sim2.0
• Case Studies and Results
• Conclusion
Conclusion

• Needs to **navigate the design-space of distributed training**
  • Large models and huge training dataset makes distributed training inevitable
  • Design space is complex: parallelism, memories, networks, etc.

• **ASTRA-sim2.0**: modeling emerging systems
  • Arbitrary parallelization strategies
  • Multi-dimensional networks at scale
  • Disaggregated memory system modeling

Thank You!

Reach out to me at:
william.won@gatech.edu

https://astra-sim.github.io

ASTRA-sim2.0 paper