1st Workshop on Inter-networking challenges for AI — CoNEXT '25

You have got a few GPUs, now what?! — Experimenting with a Nano-Cluster for Distributed Training of AI Models

Giuseppe Aceto, Ciro Guida, Valerio Persico, Antonio Pescapè

December 1st, 2025

Università degli Studi di Napoli Federico II

Motivation and Research Context

The Exponential Growth

- Deep Learning models (especially LLMs) are growing in **size and complexity**
- *Key Stat:* Compute requirements for SOTA models double every ~**3.4 months**
  - This pace far exceed hardware improvements (Moore's Law)

https://openai.com/index/ai-and-compute

The Single-Node Barrier

- Training modern models on a single device is **practically impossible**
- *The "GPT-3 Example":* Training GPT-3 (175B parameters) on a single NVIDIA V100 GPU would take approximately **288 years**

Narayanan, et al. Efficient large-scale language model training on gpu clusters using megatron-lm

The Necessary Shift

- **Scale-up** (better single GPU) is not enough.
- **Scale-out** (Distributed Training) is the mandatory standard to accelerate the process
- **Workload must be spread** across multiple nodes to achieve reasonable training times

Practical Limitations: The Accessibility Gap

Literature Gap

- **Research** focuses almost exclusively on **medium-to-hyperscale clusters**
- High-end interconnects (e.g., InfiniBand) and massive GPU counts create an *oligopoly in experimental setups*
  - This hardware is strictly **out of reach for Academia and SMEs with limited budgets**

Cloud Trade-offs

- While cloud GPU services are common, they present **distinct drawbacks**:
  - **Cost Unpredictability**: it is *"not trivial to estimate and plan"* final expenditure for new model experimentation
  - **Data Sovereignty**: privacy laws (e.g., GDPR) or security concerns often rule out public cloud solutions
  - **Operational Control**: access to remote services can be impaired

Research Questions

RQ1: Viability and Accessibility

- *To what extent can small-scale multi-GPU clusters be viable for academic research and SMEs?*
- Focus: Moving beyond hyperscale requirements to "democratic" and local model training

RQ2: Configuration and Efficiency

- *How do configuration choices affect performance?*
- Key variables analyzed:
  - Batch Size
  - Parallelism Strategies (Data, Tensor, Pipeline)
  - Cluster Setup (Inter-node vs. Intra-node)

"You've got a few GPUs, now what?!"

Background: Distributed Training

The Collaborative Goal

- Scale-out: Using multiple devices to speed up training (splitting the workload)
- Memory Expansion: Training models that don't fit on a single device (splitting the model)
- *Key Constraint:* Devices must operate collaboratively, requiring constant synchronization

What Travels on the Network?

- **Gradients**: To synchronize model updates across replicas
- **Activations**: Intermediate results passed between layers (in model parallelism)
- **Parameters**: Weights broadcasted or reduced to ensure consistency

Factors Impacting Performance

- We identify three critical factors defined by the parallelization scheme:
  - Volume: *How much* data is exchanged (MB vs GB)
  - Frequency: *How often* communication occurs (per batch vs per layer)
  - Connectivity: *Which* GPUs need to talk to each other (intra-node vs inter-node)

Background: Collective Communication Libraries

Software Abstraction Layer

- Complex synchronization logic is handled by **specialized libraries** (e.g., NCCL, MSCCL, Gloo, RCCL, ACCL, etc.)
- Goal: maximizing bandwidth utilization while **abstracting the underlying hardware topology**

Primitives: Redistribution and Consolidation

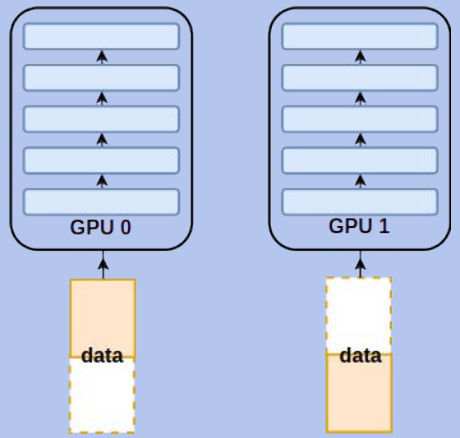
- Libraries implement standard "Collective Operations" to define data flow:
  - **1:1** (Send/Receive)
  - **1:N / N:1** (Scatter, Gather, Broadcast, Reduce)
  - **N:N** (All-Gather, All-Reduce, All-to-All)

Protocol Versatility

- Seamless support for **diverse interconnect technologies**:
  - High-End: NVLink, NVSwitch, InfiniBand (RDMA).
  - Commodity: *standard TCP/IP sockets* (enabling training on standard Ethernet)

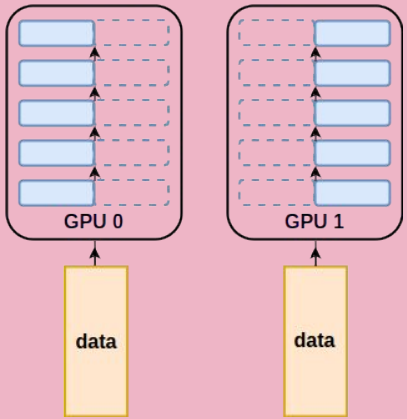
Background: Parallelization Strategies

Data Parallelism (DP)



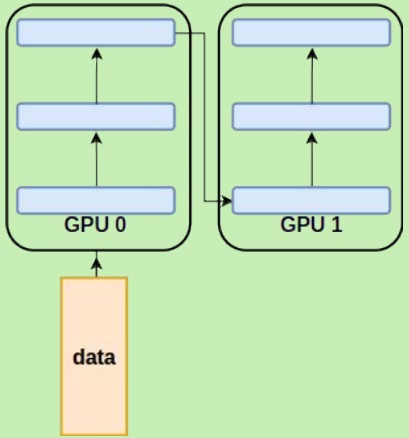
- Replicate model on each GPU
- Each processes a different batch
- Requires **gradient synchronization**

Tensor Parallelism (TP)



- Split weight matrices across GPUs
- GPUs exchange partial results for *each layer*
- Very **communication-intensive**

Pipeline Parallelism (PP)



- Split model into "stages"
- GPUs pass activations between stages
- Reduces memory demand, **moderate communication**

Background: Other Key Factors

Virtual communication topology

- Ring, tree, hierarchical, hybrid
- Determines communication paths and latency

Physical topology

- PCIe layout, GPU–GPU connectivity
- Node-to-node network layout

Interconnection technologies

- PCIe, NVLink / NVSwitch, GPUDirect
- Ethernet (1/10/25/100 GbE)
- InfiniBand, RDMA

Finding the "Best" Configuration

The challenge

Performance emerges from the **interplay** of:

- Parallelism strategy
- Collective ops
- Virtual + physical topologies
- Communication backend
- Network and GPU interconnects
- Model architecture and size

Reality check

Exploring all combinations is **not always possible:**

- Many choices are forced by available hardware
- Small labs cannot explore dozens of optimization paths
- The model size itself constraints feasible strategies

Testbed Setup

Minimalistic nano-cluster with:

- 2× Dell PowerEdge R7525 nodes
  - 2× AMD EPYC 75F3 (32 cores each), with 512 GB RAM
  - 2× NVIDIA A40 GPUs per node (48 GB VRAM each)
  - Intra-node interconnect: PCIe Gen4 ×16 (≈ 64 GB/s)
  - Inter-node network: 1 Gigabit Ethernet
- Controller node for orchestration and data collection
- **No NVLink, no InfiniBand, no RDMA**

Distributed training setup

- **1×2**: One node, two GPUs (PCIe only)
- **2×1**: Two nodes, one GPU each (network only)
- **2×2**: Two nodes, two GPUs each (hybrid PCIe + network)
- Results compared against single-GPU (1x1)

Software

- Python 3.8 + CUDA 12.1
- [1]ColossalAI v0.4.9 for distributed training
- [2]NCCL v2.26 as communication backend

[1] https://github.com/hpcaitech/ColossalAI/tree/v0.4.9
[2] https://github.com/NVIDIA/nccl/tree/v2.26.2-1

Training Setup

ResNet18

- **Type**: convolutional
- **# parameters:** ~11M parameters
- **Dataset**: CIFAR-10
- **Batch sizes**: 32 to 2048
- **Parallelism**: DP
  - Sufficient due to small model size

GPT-2

- **Type**: LLM decoder-only
- **# parameters:** (Small: 124M → XL: 1.5B)
- **Dataset**: Wikitext-2
- **Batch sizes**: 4 to 16[1]
- **Parallelism**: DP, TP, PP
  - PP used to stress our nano-cluster with larger variant of GPT-2

Architecture	#Parameters	#Layers	Peak Runtime Memory[GB]
ResNet-18	11M	18	1.37
GPT2-small	124M	12	13.68
GPT2-medium	355M	24	29.94
GPT2-large	774M	36	46.31
GPT2-XL	1500M	48	-

OOM in our commodity setup

[1] Out-of-memory with batch size of 32

Hyperparameters informed by preliminary results (~80% ResNet acc., ~20 GPT-2 ppl)

Evaluation Metrics

End-to-End Training Performance

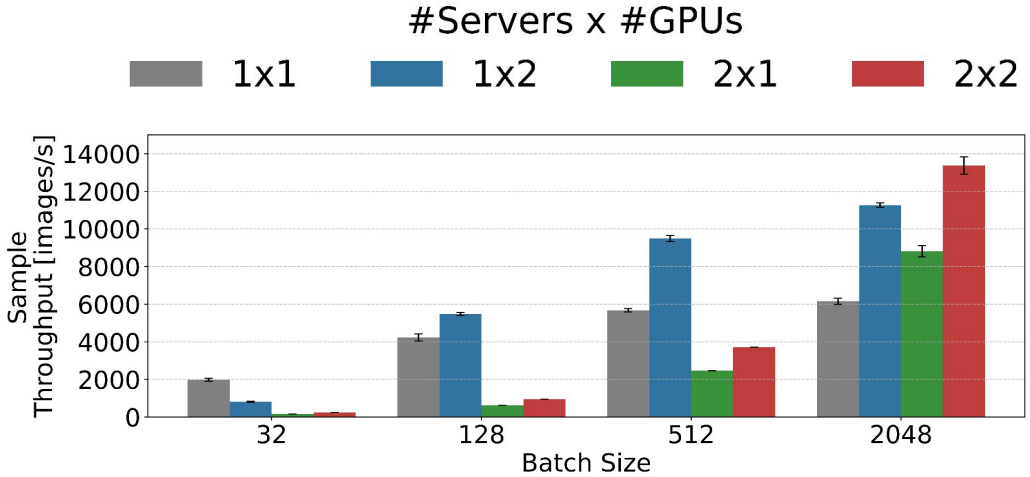
- **Sample Throughput**: Raw processing speed measured at the application layer
  - *ResNet-18:* Measured in images/s
  - *GPT-2:* Measured in tokens/s
- **Scaling factor** (S_n): Quantifies the benefit of distributing workload across GPUs
  - T_n / (n * T1)
  - S_n = 1 ideal linear scaling
  - $1 < S_n < 1/n$ communication overhead
  - $S_n < 1/n$ load distribution is detrimental

Network-Level Diagnostics

- **Context**: Critical for constrained environments (commodity 1 GbE, no RDMA/NVLink).
- **Metrics**: *Network Throughput* & *Network Volume* (Ethernet frame size)
- **Goal**: Identify bandwidth bottlenecks affecting scaling

- Experimental Protocol: All metrics averaged over **5 experimental repetitions** to mitigate runtime fluctuations.

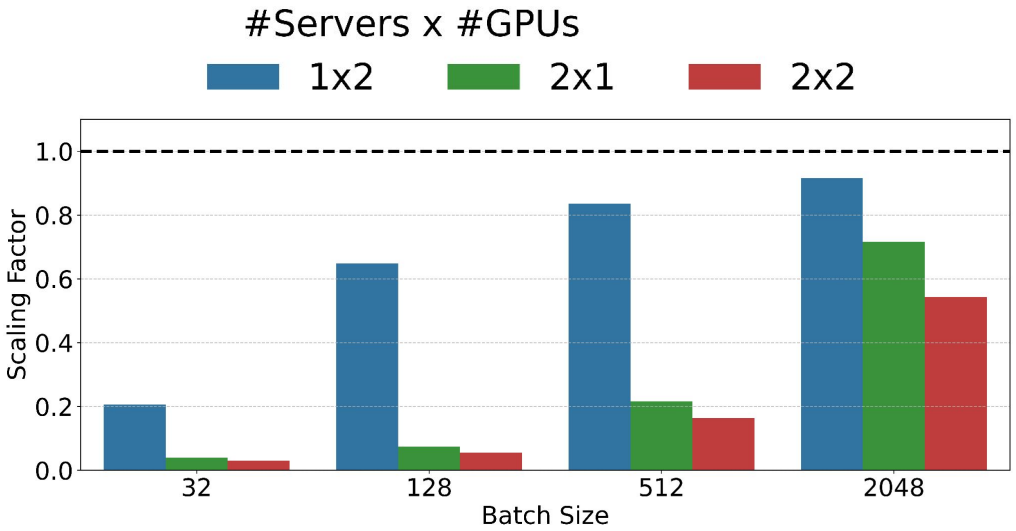
ResNet-18: Sample Throughput



- Larger batch sizes → **higher throughput**
- **1×2 and 2×2 outperform 1×1** for batch ≥128
- Peak: ≈**13k images/s in 2×2** with batch 2048
- **Small batch** (32) **performs *worse*** when distributed

Distribution helps only when the batch is large enough to amortize synchronization

ResNet-18: Scaling Factor



- **1×2** achieves the **best scaling** (up to ≈0.9)
- **2×1 and 2×2** show **degraded** scaling
  - Inter-node communication on 1 GbE dominates
- **2×2 is only ≈16% faster** than 1×2, despite double the GPUs

Crossing node boundaries severely limits returns due to the 1 GbE bottleneck

From ResNet-18 to GPT2

ResNet-18 results show

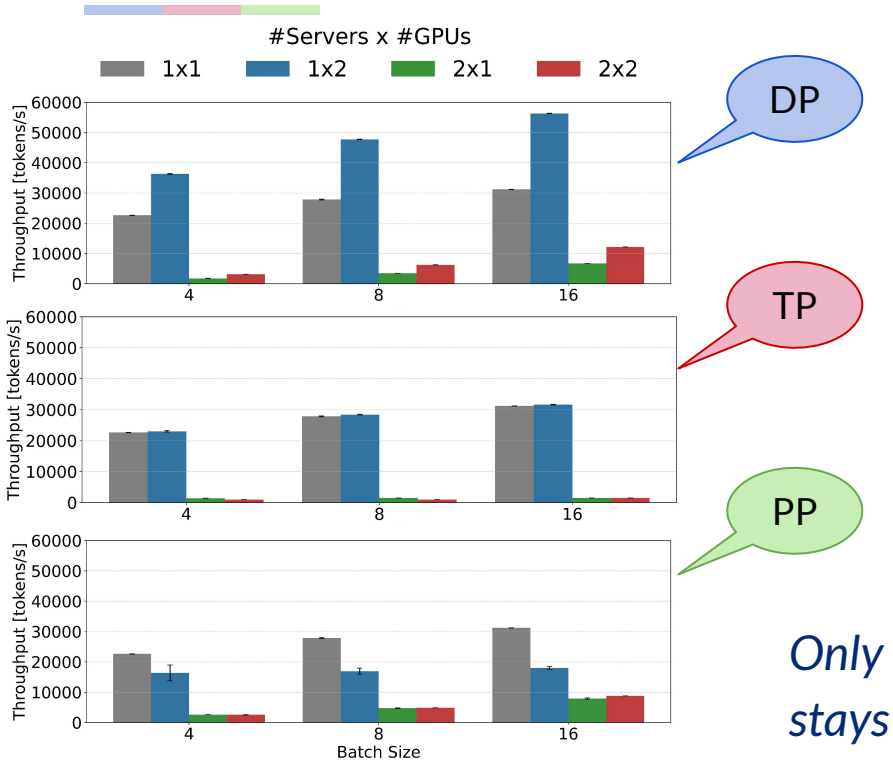
- Good scaling inside a node
- Severe degradation across nodes
- Throughput sensitive to batch size

GPT-2 brings new challenges

- Larger models
- More communication per iteration
- Higher stress on interconnects

What happens when we stress the cluster with larger models?

GPT-2 Small: Intra-node Sample Throughput



- **DP** (1×2) achieves the highest throughput: up to ≈**55k tokens/s**
- **TP** (1×2) already **slower than single-GPU** (≈26k tokens/s)
- **PP** (1×2) even slower (**≈22k tokens/s**)
  - Increasing batch size helps, but only up to 16

Only DP scales well when communication stays on PCIe

GPT-2 Small: Inter-node Sample Throughput

- **DP** (2×1 / 2×2) collapses to **6–10k tokens/s**
- **PP** (2×1 / 2×2) reaches ~8–9k tokens/s, **better than TP**
- **TP** (2×1 / 2×2) is **nearly unusable**
  - Only 1.46k tokens/s
  - Scaling factor ≈0.01–0.02

Transformer workloads expose the network bottleneck dramatically

Why Tensor Parallelism Fails

Setup	Parallelism	Sample Thrp.	Scaling Factor	Net Thrp.	Net Vol.
1×2	Data	55.01	0.93	–	–
	Tensor	25.83	0.44	–	–
	Pipeline	22.93	0.39	–	–
2×1	Data	6.73	0.11	1.63	225.40
	Tensor	1.46	0.02	1.81	1171.06
	Pipeline	7.94	0.13	1.39	174.26
2×2	Data	10.34	0.09	1.66	149.17
	Tensor	1.46	0.01	1.81	1171.79
	Pipeline	8.87	0.08	1.53	174.31

Abbreviation: Thrp. = Throughput, Net Vol. = Network Volume
Sample Thrp. (*ktokens/s*), Net Thrp. (*Gb/s*), Net Vol. (*GB*)

TP saturates the network

- Requires **exchanging partial layer results** every forward/backward step
- **Traffic volume** in 2×1 and 2×2: ≈**1170 GB per run**
- Measured throughput on 1 GbE: **1.81 Gb/s full-duplex**
- **GPUs spend more time waiting** than computing

Pipeline Parallelism: A Viable Option

Setup	Parallelism	Sample Thrp.	Scaling Factor	Net Thrp.	Net Vol.
1×2	Data	55.01	0.93	–	–
	Tensor	25.83	0.44	–	–
	Pipeline	22.93	0.39	–	–
2×1	Data	6.73	0.11	1.63	225.40
	Tensor	1.46	0.02	1.81	1171.06
	Pipeline	7.94	0.13	1.39	174.26
2×2	Data	10.34	0.09	1.66	149.17
	Tensor	1.46	0.01	1.81	1171.79
	Pipeline	8.87	0.08	1.53	174.31

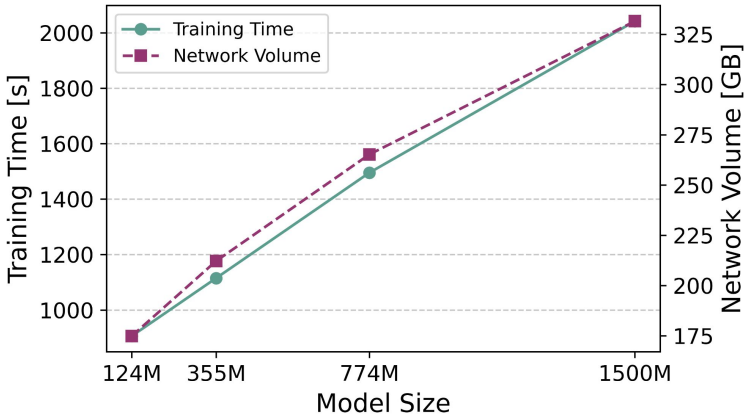
Abbreviation: Thrp. = Throughput, Net Vol. = Network Volume
Sample Thrp. (*ktokens/s*), Net Thrp. (*Gb/s*), Net Vol. (*GB*)

PP adapts better to limited bandwidth

- Stable throughput in 2×1 and 2×2: **≈7–9k tokens/s**
- **Network volume moderate**: ≈174 GB per run
- **Network throughput**: **2×2 ~10% better than 2×1** due to reduced pipeline bubbles (one-forward-one-backward scheduling)

PP works better than DP in low-bandwidth setup

Scaling to Larger GPT-2 Variants



- **Linear Scaling**: Training time and network traffic **grow linearly** with model complexity, ensuring cost predictability
- **Overcoming Memory Limits**: **PP** makes training 1.5B-parameter models **viable on memory-constrained hardware**
- **Network Usage**: Volume increases proportionally (174 GB to 330 GB), leaving the **bottleneck at the limited bandwidth** of 1 GbE interconnect

Despite the efficiency of the parallelization strategy, the 1 GbE interconnect remains the dominant saturation point, limiting the overall training speed

Key Takeaways (1/2): What We Learned About Scaling

Scaling only works *inside* a node

- **1×2** gives near-ideal scaling (up to ~0.9)
- As soon as the **network** is involved → **strong degradation**
- **2×2** only **16% faster than 1×2**, despite 2× GPUs

Network quickly becomes the bottleneck

- **1 GbE is insufficient** for multi-node training
- Communication time > computation time

Key Takeaways (2/2): Transformers Stress Everything

DP, TP, PP behave very differently

- **DP**: best performance, but limited by GPU memory
- **TP**: saturates 1 GbE → unusable
- **PP**: only viable strategy for large models

Transformers amplify bottlenecks

- Larger models → **more synchronization**
- TP network volume ≈ **1170 GB**
- **PP remains usable** but still network-bound

On constrained interconnects, only a subset of strategies is feasible

Future Directions

Addressing Network Bottlenecks

- **Network Upgrade**: Evaluating affordable high-speed technologies (e.g., 10 GbE) to mitigate the observed bandwidth saturation
- **Physical Topology**: Experimenting with Multi-NIC setups (one NIC per GPU) to optimize inter-node traffic distribution

Optimizing Training Efficiency

- **Advanced Parallelism**: Integrating ZeRO (Zero Redundancy Optimizer) to minimize memory redundancy and communication overhead
- **Data Strategy**: Analyzing dataset subsampling trade-offs to balance training speed vs. model accuracy

Long-term Vision

Defining cost-effective guidelines for **SMEs and Academia** to train large models on constrained "Nano-Clusters"

Thanks for your attention!

For references and details check the **paper**:
https://dl.acm.org/doi/10.1145/3769695.3771676

Icons by Flaticon (https://www.flaticon.com)