LLMs pretraining: From Distributed to Decentralized setting

Ferdinand Mom, Research Engineer @Huggingface

DP2E-AI 2025 workshop

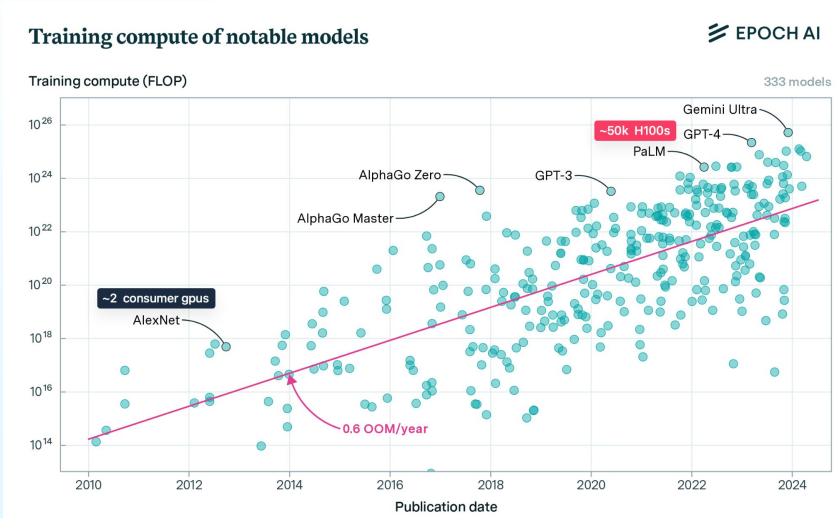
Previously video compression @Interdigital @VLC media player (Google Summer of Code)

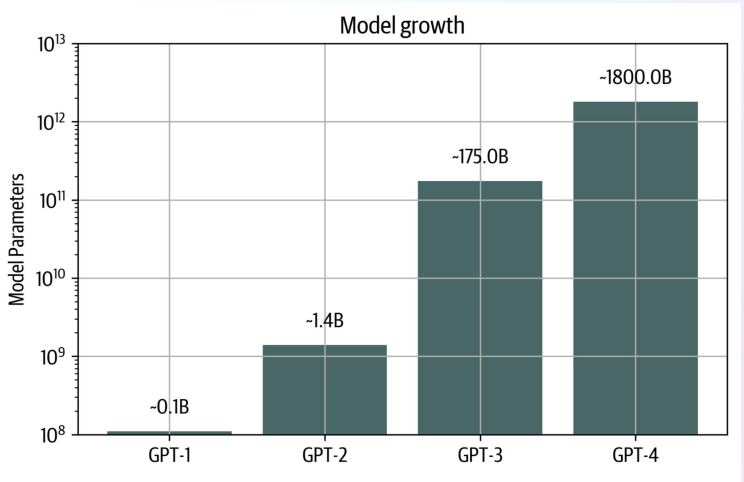
Now Pre-training @Huggingface: Picotron, Nanotron, The Ultra-Scale Playbook: Training LLMs on GPU Clusters

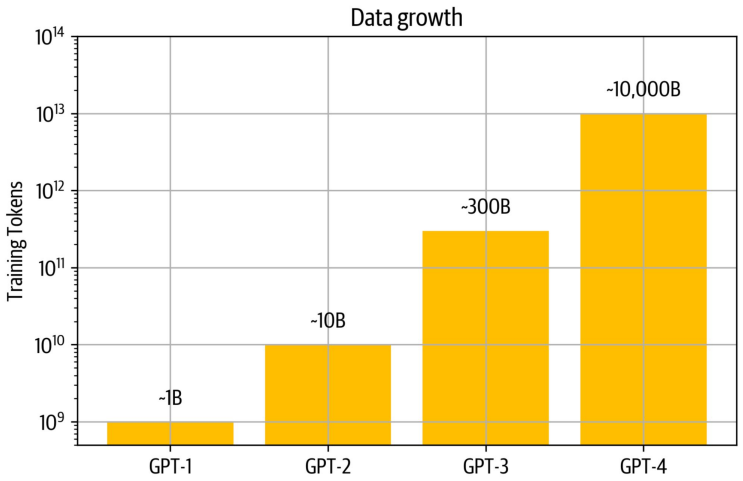
@3outeille @FerdinandMom

Why do we need distributed training ?

Scaling laws tell us that model performance is expected to scale up with larger **model size (number of parameters)**, **more compute (FLOPs)** and more **training data (tokens)**





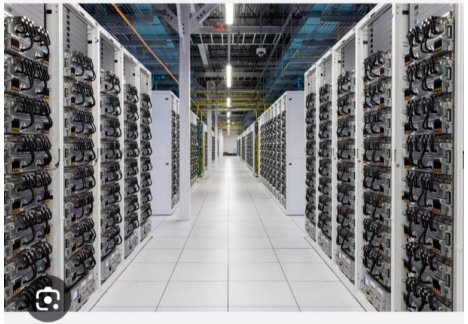


Why do we need distributed training ?

Breaking news: In the past 24 hours, details were leaked about GPT-4. The information was analysis by Dylan Patel posted here on SemiAnalysis, but put juicy details behind a paywall. Yam Peleg shared those details on Twitter, but then took down his tweet thread "due to a copyright claim."

However, his information is still available here, and we will summarize what we know about GPT-4 and what it means. Details are below, but at the top-line, we know:

- GPT-4 is a mixture-of-experts model, with 16 experts of 111B parameters each.
- It took about 2 x 10^25 FLOPS to train, with 13 trillion token (passes).
- Estimated pre-training hardware utilization cost of $63 million, using 25,000 A100s almost 100 days to do the training.
- The training and architecture was to optimize it for inference, and inference costs were about 3 times that of GPT-3 / DaVinci.



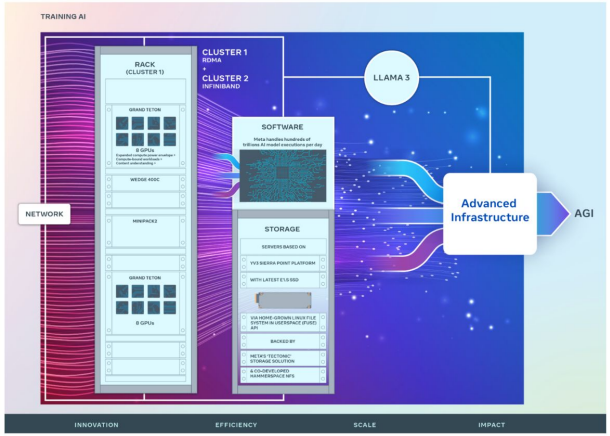
Inside the 100K GPU xAI Colossus Cluster that...

Visit >

Creator: patrick kennedy pa…

Want to know where this information comes from? **Learn more**

Images may be subject to copyright. **Learn More**



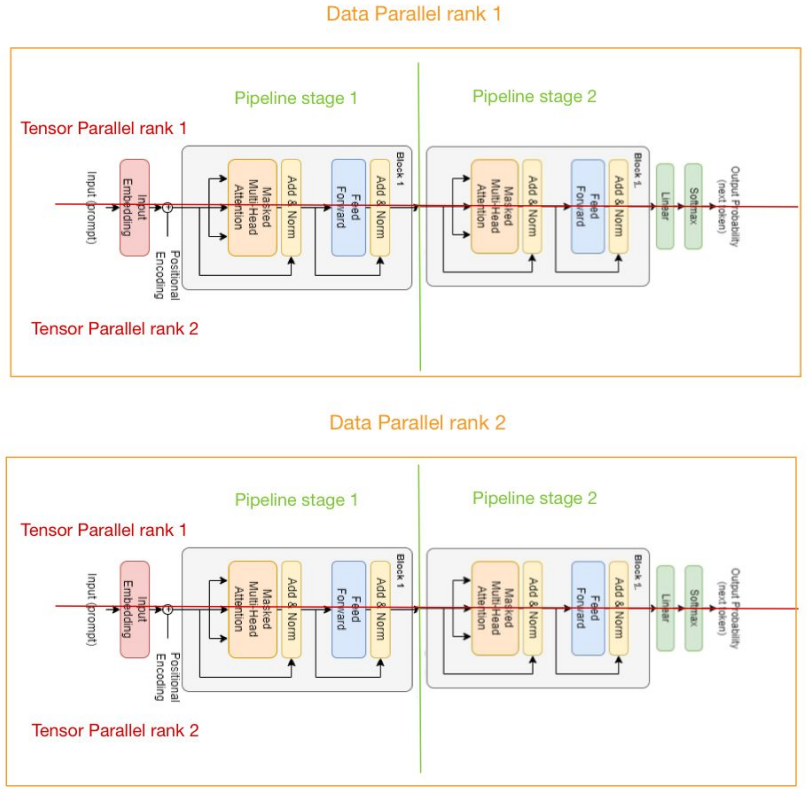
Under the hood

Our newer AI clusters build upon the successes and lessons learned from RSC. We focused on building end-to-end AI systems with a major emphasis on researcher and developer experience and productivity. The efficiency of the high-performance network fabrics within these clusters, some of the key storage decisions, combined with the 24,576 NVIDIA Tensor Core H100 GPUs in each, allow both cluster versions to support models larger and more complex than that could be supported in the RSC and pave the way for advancements in GenAI product development and AI research.

What is distributed training ?

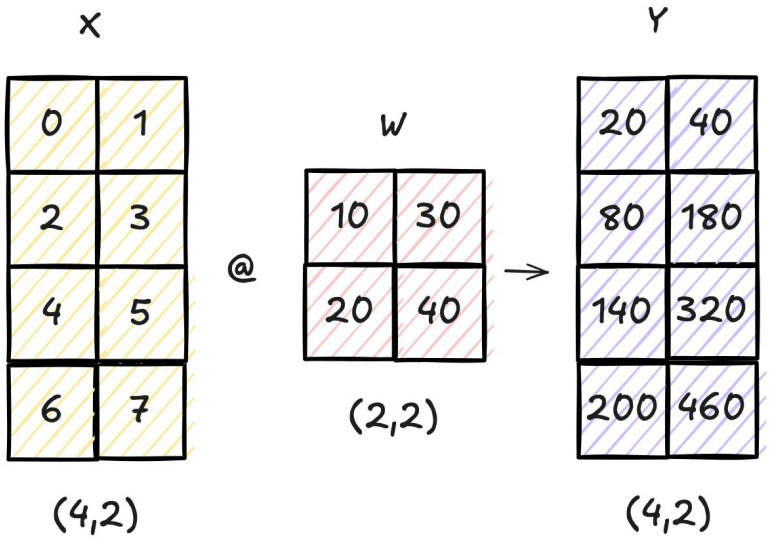
As model get larger, it cannot fit within a **single GPU**.

Model need to be **distributed across GPUs along different axes**



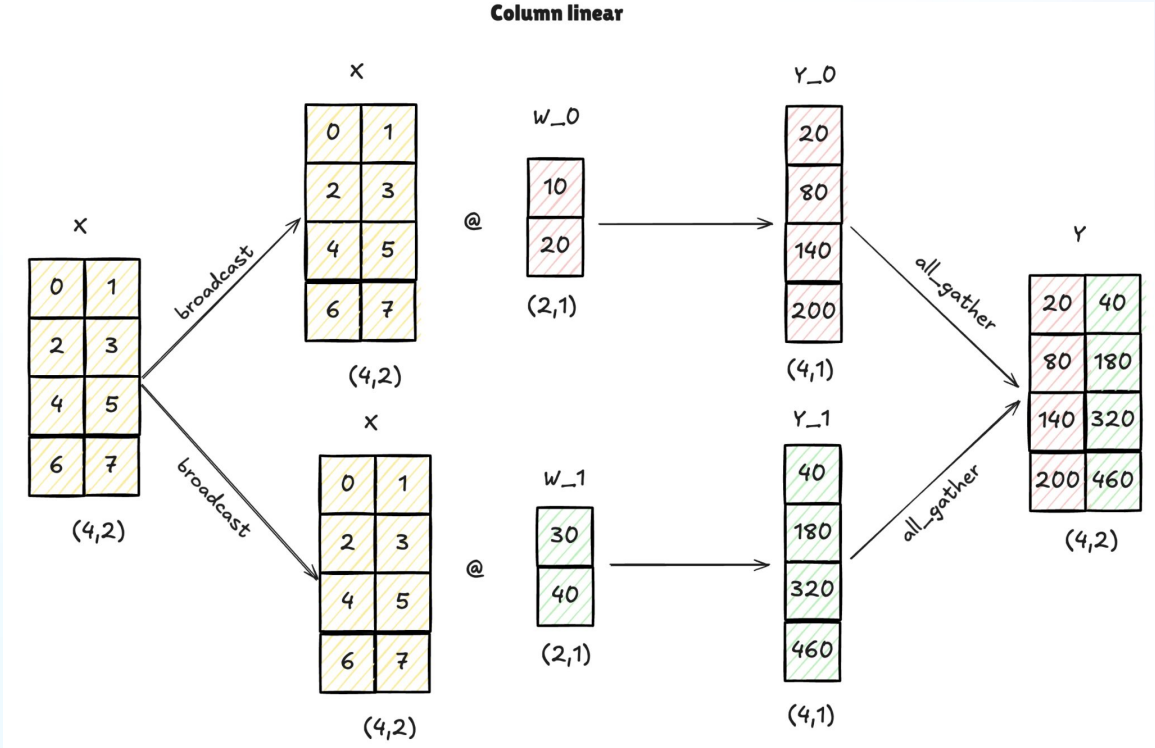
Tensor Parallel

What if the model doesn't fit ? Split matrix multiplication



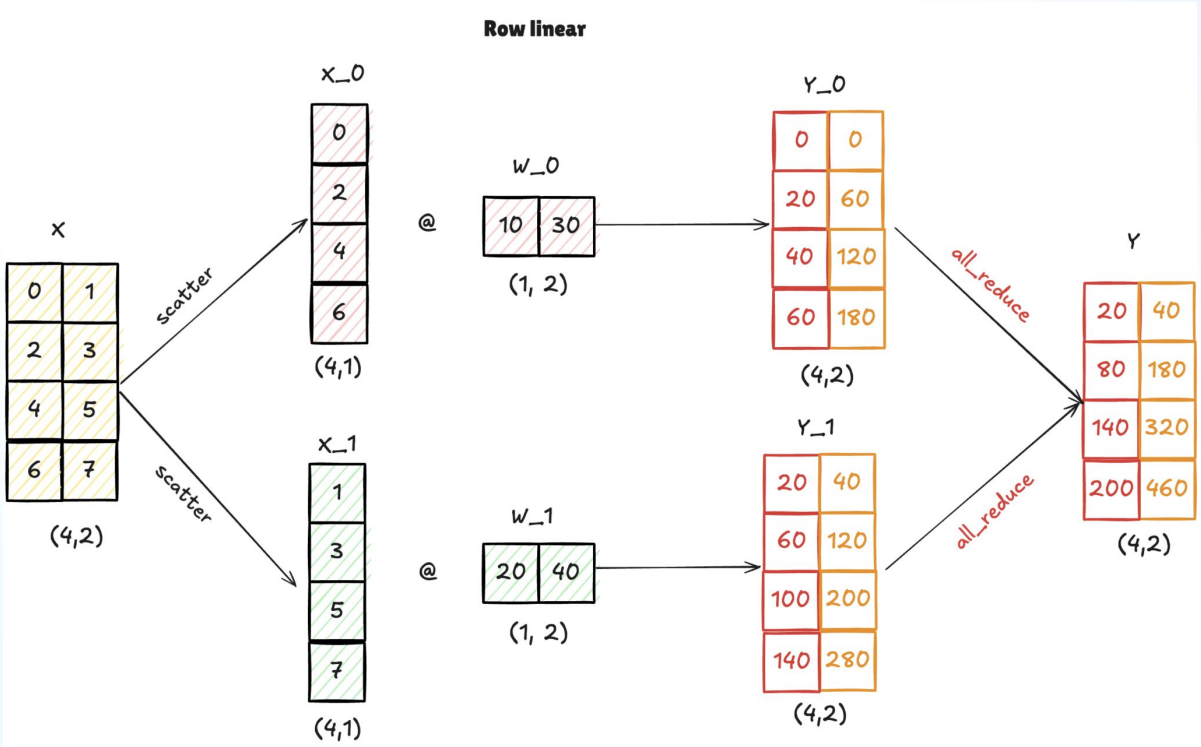
Tensor Parallel

What if the model doesn't fit ? Split matrix multiplication



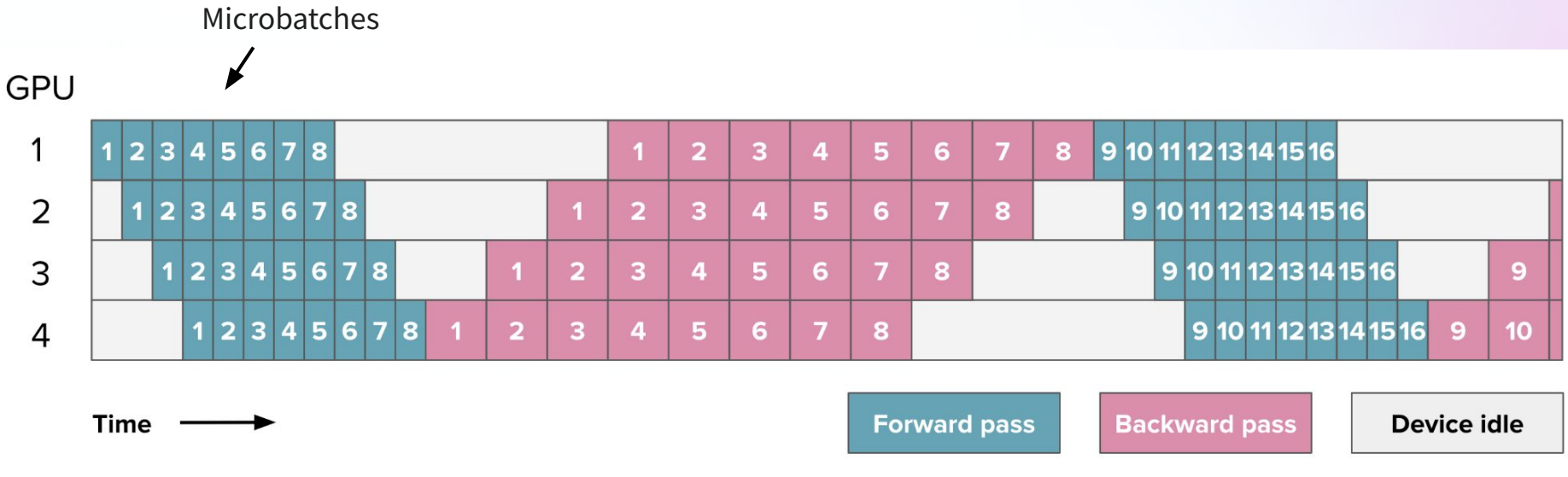
Tensor Parallel

What if the model doesn't fit ? Split matrix multiplication



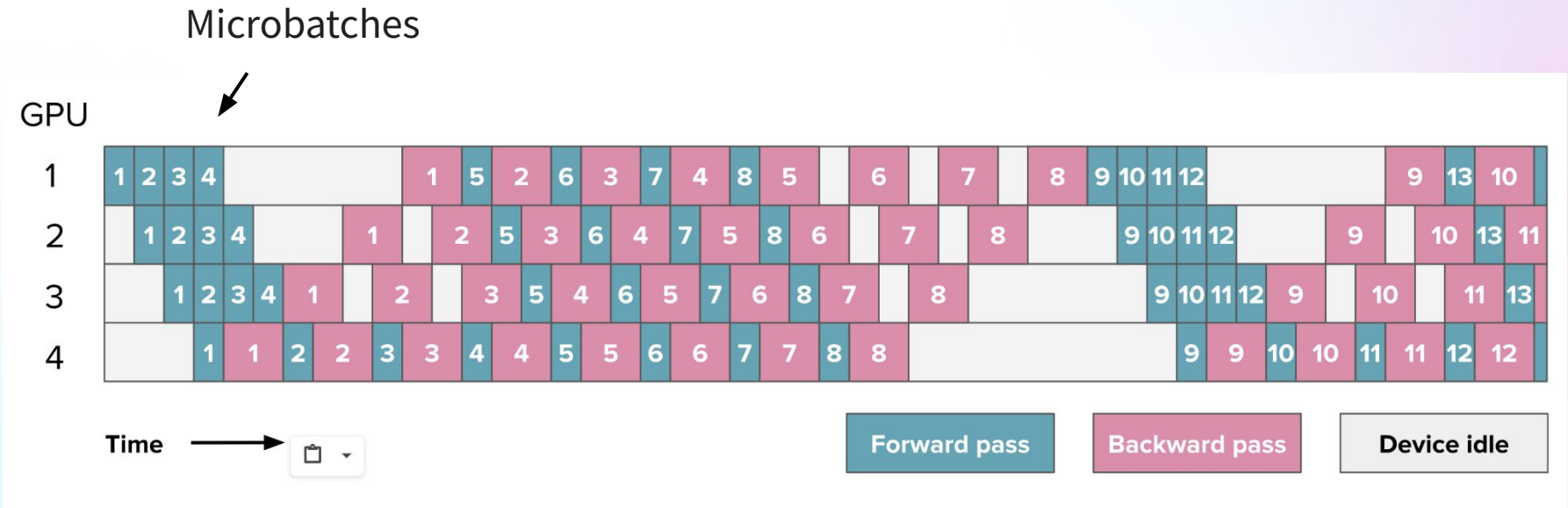
Pipeline Parallel

Share layer across GPUs !
AFAB - All Forward All Backward

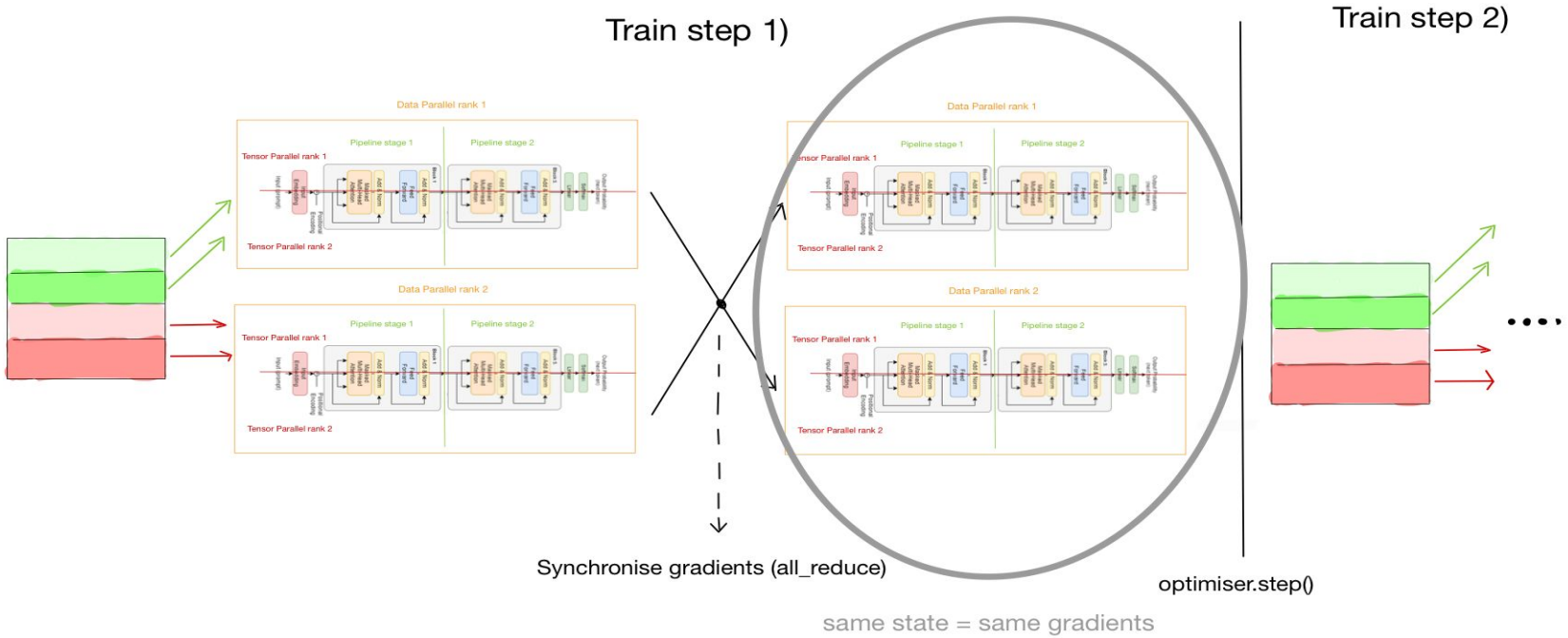


Pipeline Parallel

1F1B: 1 Forward 1 Backward



Data Parallel



Under the hood of distributed training

all_reduce is central in deep learning workload !

Data parallel => **all_reduce (at the end of every training step)**

Tensor parallel => **all_reduce (every linear layers) +** all_gather (just 1 time at final linear layer)

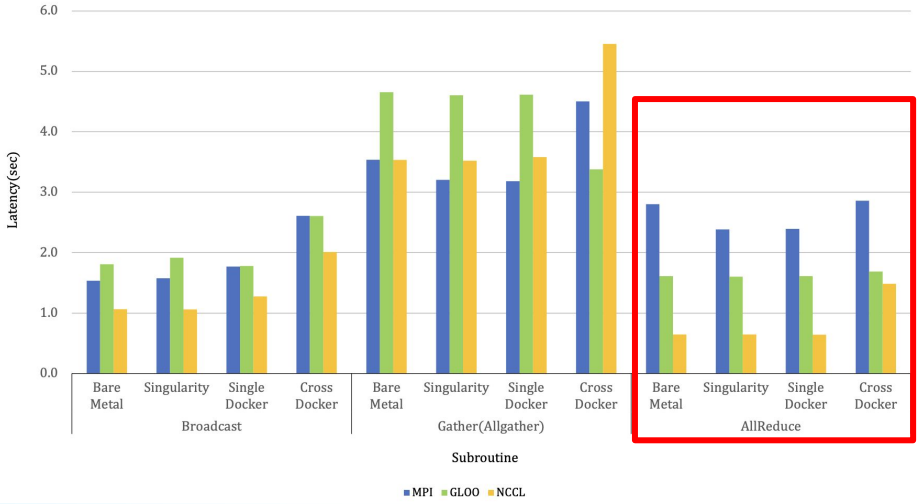
Pipeline parallel => send/recv (after every layers)

Backend	gloo		mpi		nccl	
Device	CPU	GPU	CPU	GPU	CPU	GPU
send	✓	✗	✓	?	✗	✓
recv	✓	✗	✓	?	✗	✓
broadcast	✓	✓	✓	?	✗	✓
all_reduce	✓	✓	✓	?	✗	✓
reduce	✓	✗	✓	?	✗	✓
all_gather	✓	✗	✓	?	✗	✓
gather	✓	✗	✓	?	✗	✓
scatter	✓	✗	✓	?	✗	✓
reduce_scatter	✗	✗	✗	✗	✗	✓
all_to_all	✗	✗	✓	?	✗	✓
barrier	✓	✗	✓	?	✗	✓

From MPI to NCCL

Table 1. Hardware overview of experimental system.

	Experimental Server
GPU	4 NVIDIA GeForce RTX 3080 GPUs (12 GiB)
CPU	1 Intel Core i9-10900 processor (10 cores)
CPU Memory	32 GB 2933 MHz DDR4
PCIe	bidirectional 16 GBps PCIe (Gen 3)

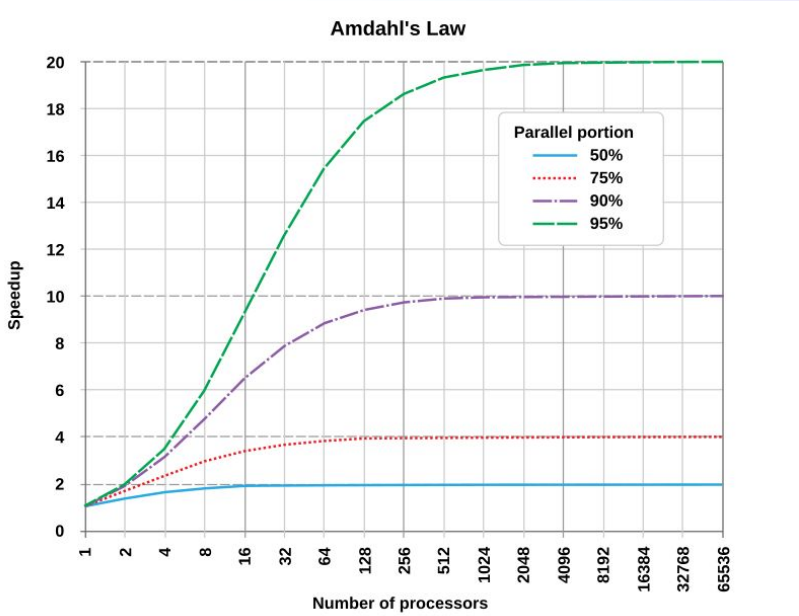


https://www.mdpi.com/2076-3417/14/12/5100

Limitations of Distributed training

Theoretically:

Amdahl's law => the speedup from **adding more chips** to a workload has **diminishing returns when** there is a **lot of synchronous activity**



Limitations of Distributed training

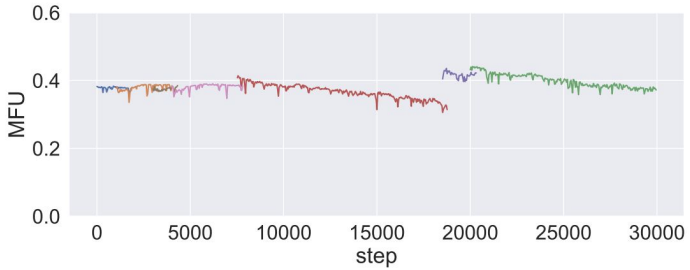


Figure 6: Inconsistent MFU observed in large-scale training. Different colors denote distinct executions of the same training job.

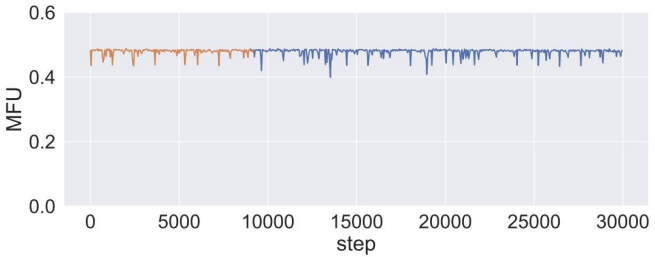


Figure 12: The MFU becomes stable after addressing the stragglers and problematic code segments. Different colors represent different training trials with the same setup.

Practical limit: 25% decrease in Model FLOPs Utilization (MFU) means

1. XAI cluster => 100k H100
2. (100,000 × 0.25 = 25,000) => **Equivalent of having 25k GPUs idle during this time**

GPU idle cost estimation:

- High-end AI training GPUs (like NVIDIA H100) cost ~40,000$ each
- **25,000 × 40,000$ = 1$ billion**

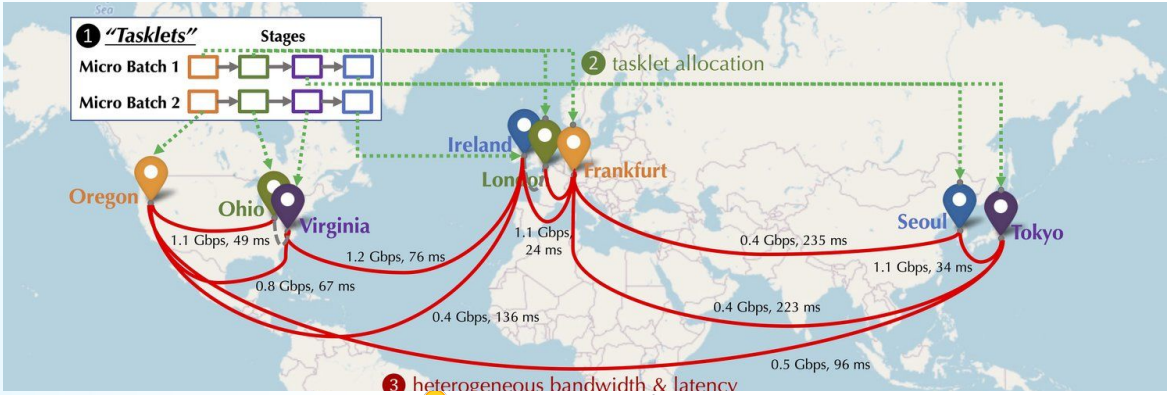
Motivation for Decentralized training

Current Problem: Not everyone can have access to compute **gathered in single place**. Even if it is the case, you have **diminishing returns**

Key idea: What if we can **gather cheap spot instances across the world** and **connect them together**. This enables us to do a 2-phase approach where we:

- Find the point where it is **optimal to scale within cluster/locally** to get best/stable performance (**distributed training**)

- **expand the same setting to other clusters** (**decentralized training**).



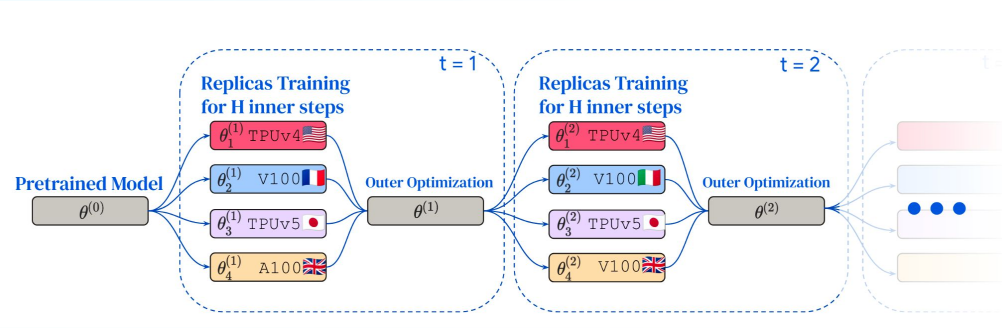
Decentralized training problems

3 key questions:

- How do we actually **“expand” setting to other clusters** ?
- How do we address **bandwidth limitations** in geographically dispersed scenarios? (Intra-cluster **~ Tbs** and Inter-cluster **~ 1 Gbs**)
- What about **fault tolerance** ?

Data parallel approach: Diloco

- Bandwidth is an issue ? Instead of communication every steps, **let's communicate only every T steps**
- **Several replicas** are converging on their own (inner loop) + **averaging in the parameter space** (outer loop)
- **Reduce communication up to 500x**
- Proven to work from **1B to 10B scales** (PrimeIntellect)



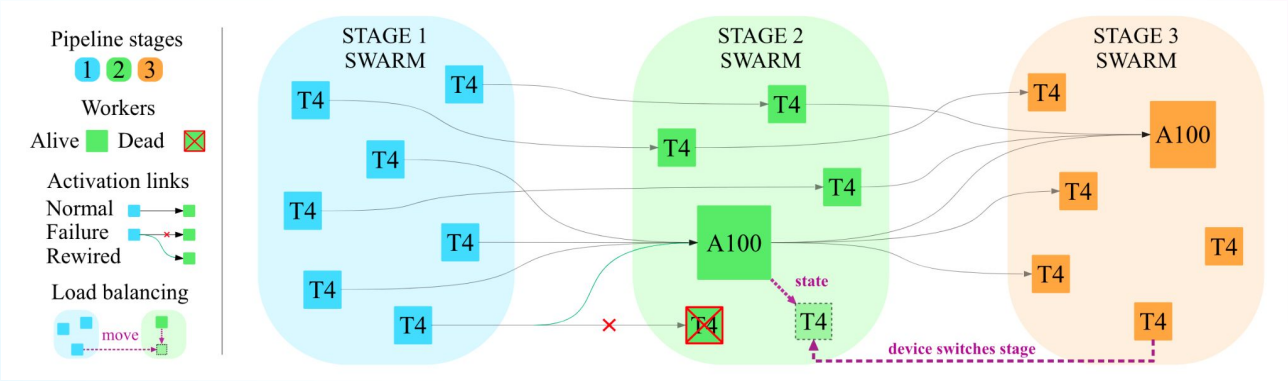
Algorithm 1 DiLoCo Algorithm

Require: Initial model $\theta^{(0)}$
Require: k workers
Require: Data shards $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$
Require: Optimizers `InnerOpt` and `OuterOpt`

for `outer step` $t = 1 \ldots T$ **do**
  for `worker` $i = 1 \ldots k$ **do**
    $\theta_i^{(t)} \leftarrow \theta^{(t-1)}$
    for `inner step` $h = 1 \ldots H$ **do**
      $x \sim \mathcal{D}_i$
      $\mathcal{L} \leftarrow f(x, \theta_i^{(t)})$
      ▷ Inner optimization:
      $\theta_i^{(t)} \leftarrow \texttt{InnerOpt}(\theta_i^{(t)}, \nabla_{\mathcal{L}})$
    end for
  end for
  ▷ Averaging outer gradients:
  $\Delta^{(t)} \leftarrow \frac{1}{k} \sum_{i=1}^{k} (\theta^{(t-1)} - \theta_i^{(t)})$
  ▷ Outer optimization:
  $\theta^{(t)} \leftarrow \texttt{OuterOpt}(\theta^{(t-1)}, \Delta^{(t)})$
end for

Model Parallel approach: Swarm Parallelism

- **Square-Cube Law** of Distributed Training => as LLMs get larger, **more time is spent doing computation compare to communication (waiting for data)**
- Cannot apply traditional Pipeline parallel directly => **Fault tolerance** (through replicas) + **load balancing within stage** (Stochastic Wiring) and **across stages** (Adaptive Rebalancing) for **optimal throughput**
- **Compression of activations and gradients**
- Possible to train with **80Mbps** at **8B** scale **with no convergence degradation** (Pluralis Research)



Compression approach: PowerSGD

- **Gradients are huge to send** => Need a way to compress gradient while maintaining training accuracy
- **Compress gradient using low-rank approximation** => we **send factorized version** that is **cheap to compute** (Power Iteration instead of SVD)
- **Compression is linear** => **compatible with all_reduce**
- Include **Error Feedback** => Keeps track of compression errors to maintain accuracy
- **10 - 100x communication reduction**

Algorithm 2 Distributed Error-feedback SGD with Momentum

hyperparameters: learning rate γ, momentum parameter λ
initialize model parameters $\mathbf{x} \in \mathbb{R}^d$, momentum $\mathbf{m} \leftarrow \mathbf{0} \in \mathbb{R}^d$, replicated across workers
at each worker $w = 1, \ldots, W$ **do**
  initialize memory $\mathbf{e}_w \leftarrow \mathbf{0} \in \mathbb{R}^d$
  for each iterate $t = 0, \ldots$ **do**
    Compute a stochastic gradient $\mathbf{g}_w \in \mathbb{R}^d$.
    $\Delta_w \leftarrow \mathbf{g}_w + \mathbf{e}_w$ ▷ Incorporate error-feedback into update
    $\mathcal{C}(\Delta_w) \leftarrow \text{COMPRESS}(\Delta_w)$
    $\mathbf{e}_w \leftarrow \Delta_w - \text{DECOMPRESS}(\mathcal{C}(\Delta_w))$ ▷ Memorize local errors
    $\mathcal{C}(\Delta) \leftarrow \text{AGGREGATE}(\mathcal{C}(\Delta_1), \ldots, \mathcal{C}(\Delta_W))$ ▷ Exchange gradients
    $\Delta' \leftarrow \text{DECOMPRESS}(\mathcal{C}(\Delta))$ ▷ Reconstruct an update $\in \mathbb{R}^d$
    $\mathbf{m} \leftarrow \lambda \mathbf{m} + \Delta'$
    $\mathbf{x} \leftarrow \mathbf{x} - \gamma (\Delta' + \mathbf{m})$
  end for
end at

3 axes of decentralized training

Prime Intellect: Decentralized pre-training **with data parallel approach**
Pluralis Research: Decentralized pre-training **with model parallel approach**
Nous Research: Decentralized pre-training **with compression approach**

They are **complementary approach in a sense that they act on different axis**, one can combine them together !

“NCCL” over public internet ?

Intra-cluster -> NCCL

Inter-cluster -> Gloo

- Not designed for WAN operation; **requires VPN** for public internet use -> **overhead**
- **Limited fault tolerance**, node failures typically require restarting the entire job
- **No built-in support for concurrent collective operations**
-

Some premises …

Prime Collective Communications Library - Technical Report

Michael Keiblinger
Prime Intellect

Mario Sieg
Prime Intellect

Jack Min Ong
Prime Intellect

Sami Jaghouar
Prime Intellect

Johannes Hagemann
Prime Intellect

“NCCL” over public internet ?

Table 12: Comparison of reduce time between PCCL and Gloo

Experiment	PCCL Time (s)	Gloo Time (s)	Improvement (%)
1. North America + Europe	90.50 ± 0.35	94.44 ± 1.84	4.15%
2. North America	35.20 ± 0.31	37.58 ± 0.85	6.33%
3. Europe	8.30 ± 0.33	9.67 ± 0.77	14.17%

Table 13: Comparison of effective throughput between PCCL and Gloo

Experiment	PCCL eff. TPT (MB/s)	Gloo eff. TPT (MB/s)
1. North America + Europe	11.85	11.37
2. North America	30.48	28.82
3. Europe	129.20	113.66

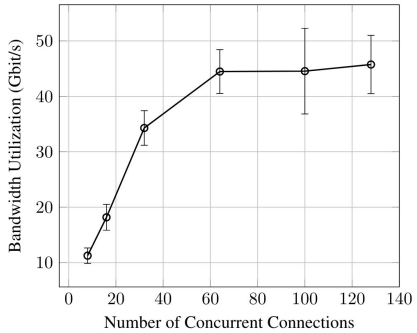
It should be noted that Gloo does not natively support concurrent all-reduce operations and can thus not effectively utilize the most crucial trick to maximize throughput over the public internet.

6.5.1 Experiment 3.1: Concurrent Connections Europe West

When using multiple concurrent all reduce operations and dispatching to multiple connections, we observe the following scaling in the setting of the Europe-West experiment:

Table 10: Summary: All-Reduce Performance (Europe West, 6 nodes)

# CON	Time (s ± s)	eff. TPT (GB/s)	TX+RX BW (Gbit/s)	TX+RX/peer (GB)
128	5.066 ± 0.560	1.694	45.74 ± 5.265	28.63
100	4.136 ± 0.757	1.622	44.55 ± 7.733	22.36
64	2.596 ± 0.233	1.655	44.47 ± 3.964	14.31
32	1.684 ± 0.167	1.275	34.29 ± 3.127	9.21
16	1.602 ± 0.219	0.669	18.17 ± 2.321	5.04
8	1.292 ± 0.159	0.415	11.25 ± 1.398	3.11



Boom project: decentralized training over national clusters

End goal:

- Releasing small model to show that it is feasible to train across clusters
- After several iterations, gathering more compute to actually train a big model

Let’s keep in touch and follow our team at hf.co/science