INTELLECT-1 Technical Report

Authors: Jaghouar et al.

Presenters: Aurélien Bondis, Parsa Toopchinezhad

Presentation Outline

1- Introduction

2- PRIME Framework

3- INTELLECT-1 Training

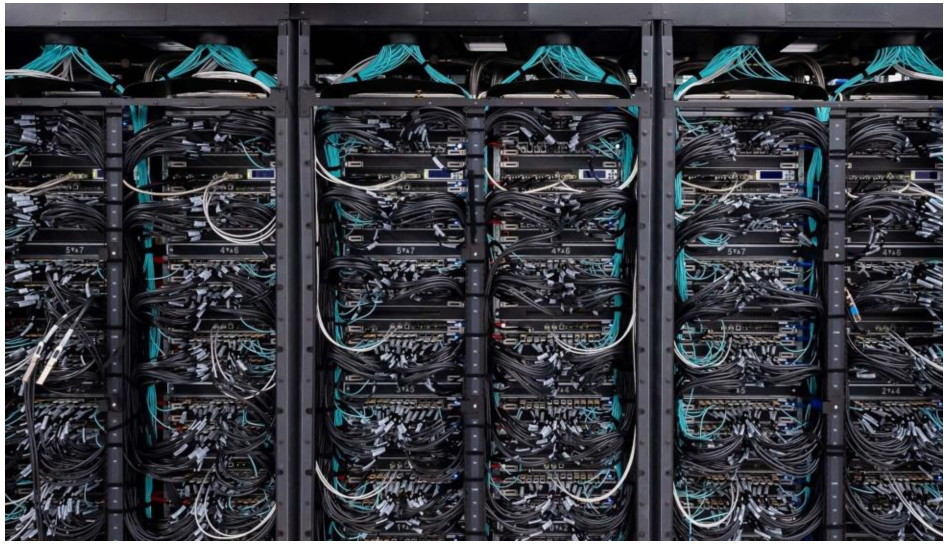
4- Conclusion & Future Work

5- Questions

Introduction

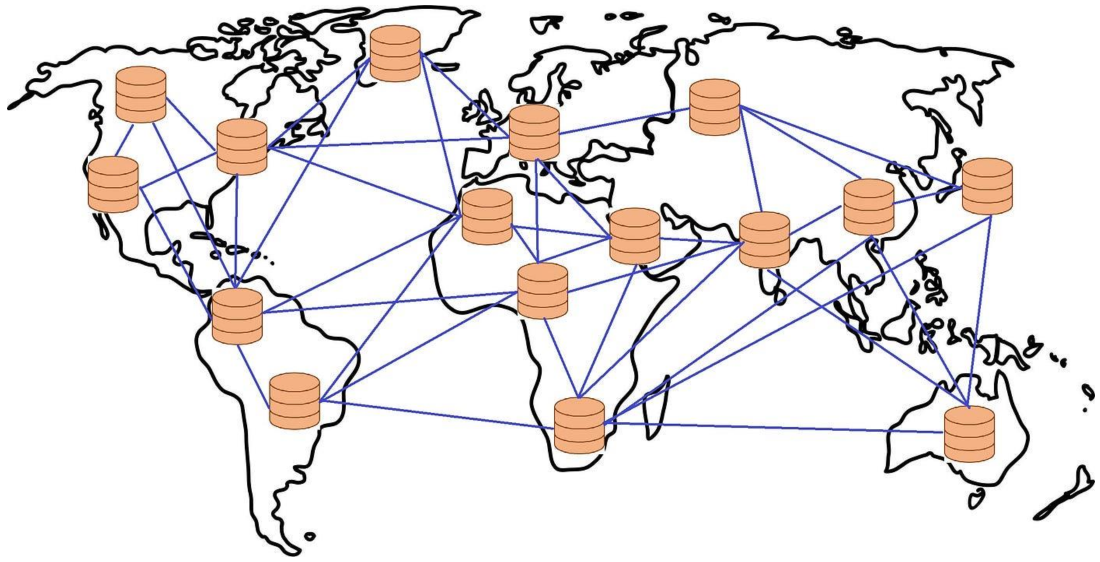
Motivation

- To train an LLM from scratch you need:
  - The algorithms
  - The data
  - The hardware
- The main limiting factor is the hardware
- Is it even possible for anyone else to compete?
- With decentralized training, we might stand a chance!



Challenges

- Typical Internet throughputs are x1000 times slower than HPC environments
- Nodes can join and leave at any time
- Nodes have heterogenous hardware
- *Much more…*



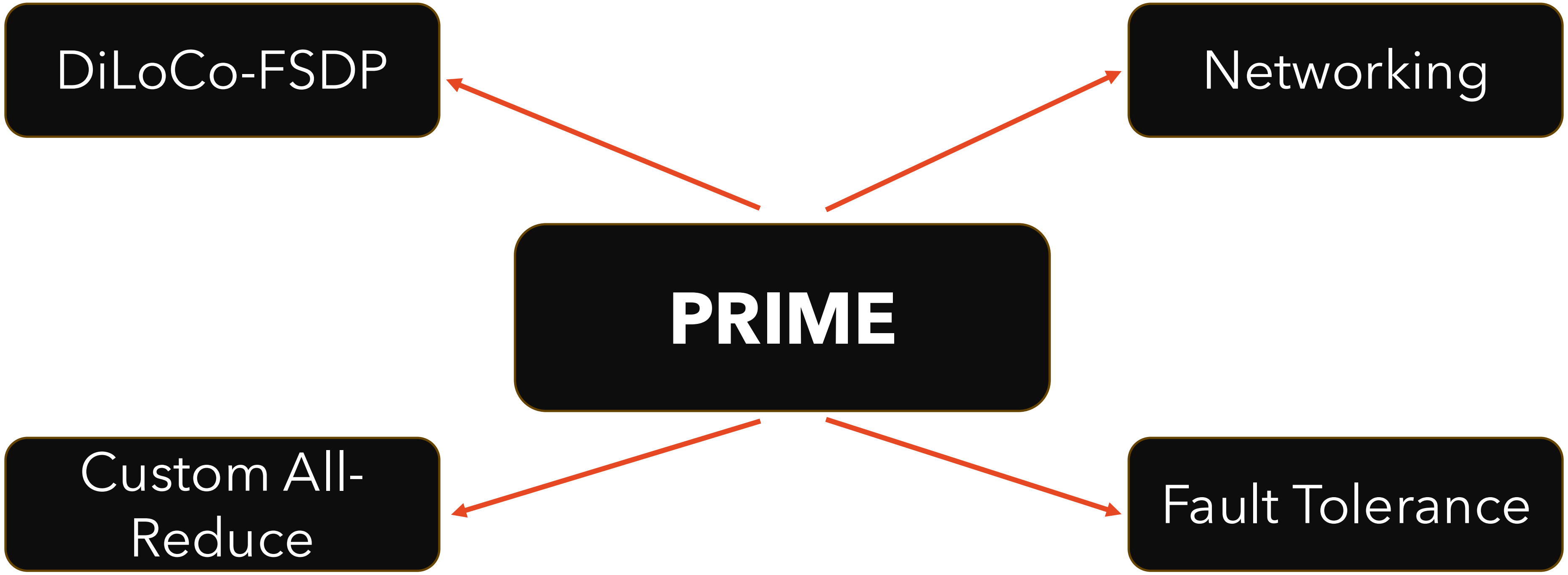
Contributions

- **INTELLECT-1**: the first 10 billion parameter language model collaboratively trained across the globe
- **PRIME**: a framework for enabling decentralized training
- Solving many algorithmic and engineering challenges along the way

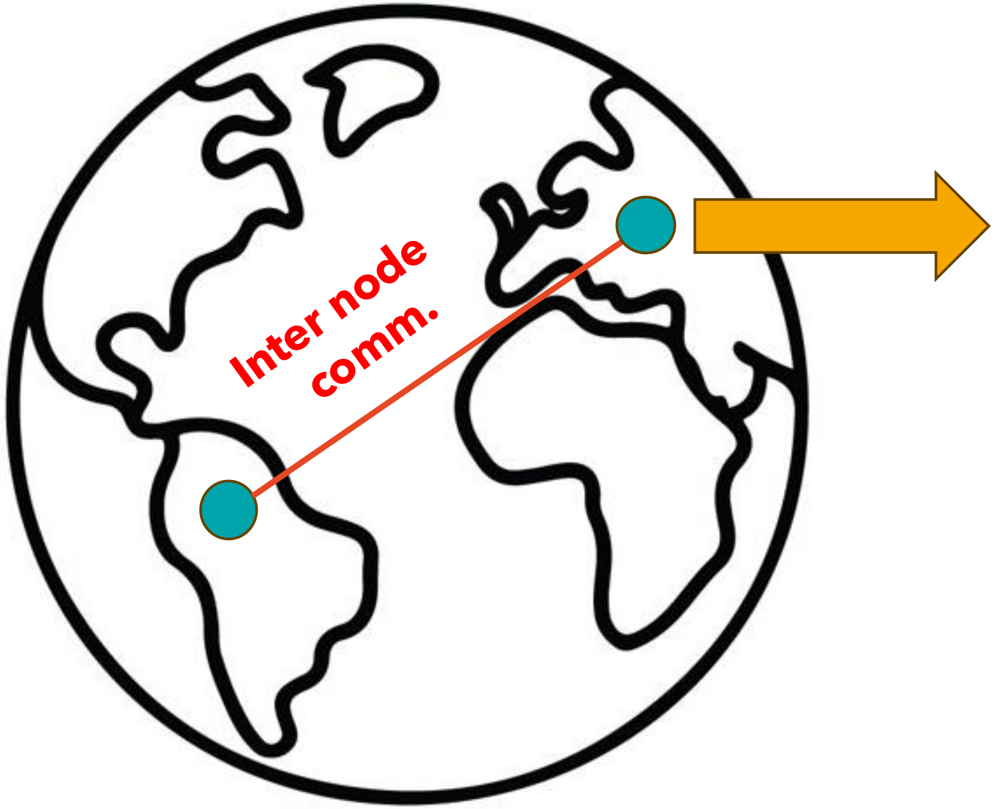
Training Dashboard

PRIME Framework

PRIME: A Scalable Distributed Training Framework



Training Setup



Intra node comm.

DiLoCo: Inter-Node Opt.

- Goal: Communicate as little as possible
- PRIME uses OpenDiLoCo
- Each node does *h* inner optimization steps
- Generally, h = 500 --> x500 less communication

Algorithm 1 DiLoCo Algorithm

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

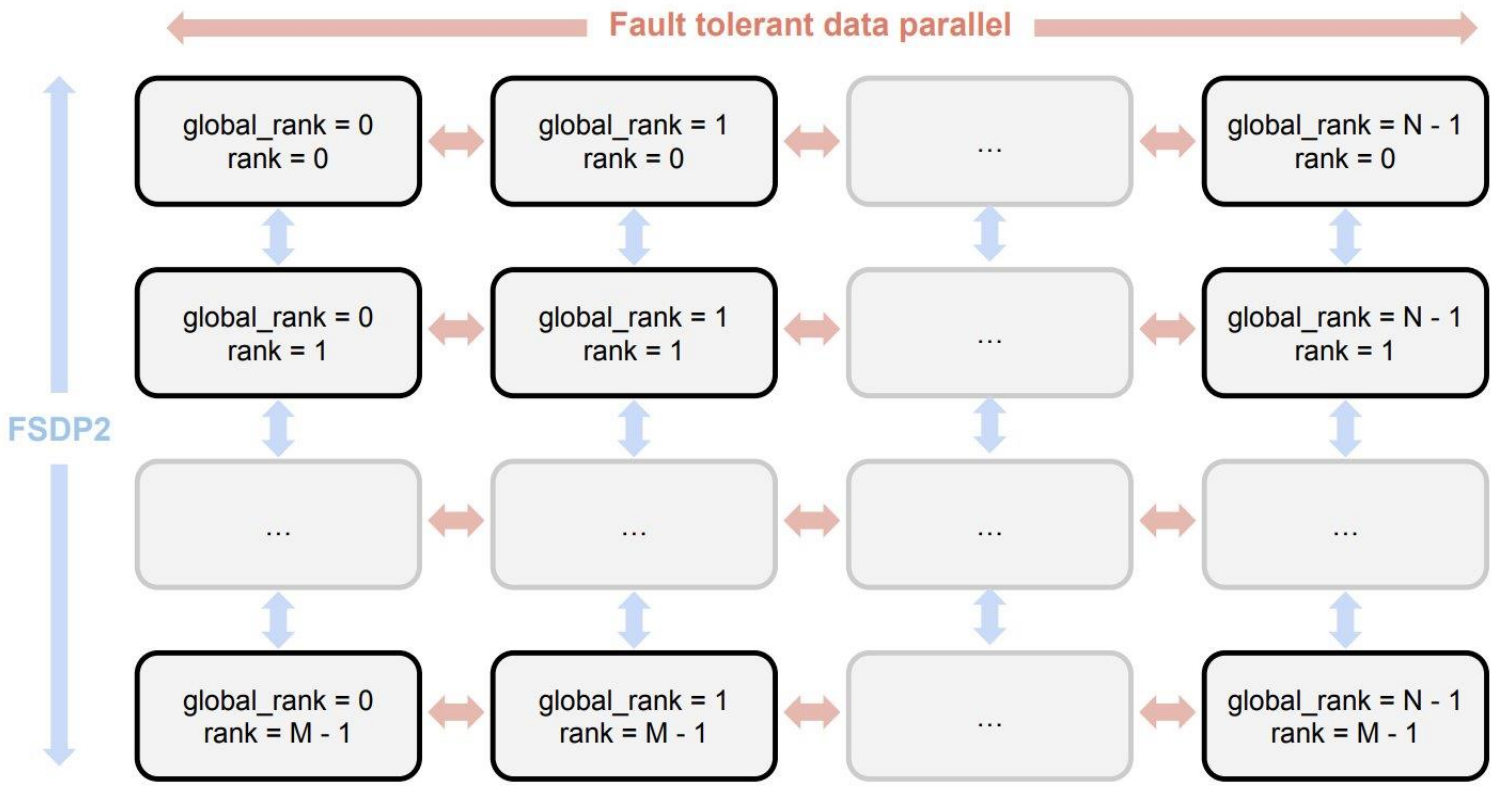
for outer step $t = 1 \ldots T$ **do**
    for worker $i = 1 \ldots k$ **do**
        $\theta_i^{(t)} \leftarrow \theta^{(t-1)}$
        for inner step $h = 1 \ldots H$ **do**
            $x \sim \mathcal{D}_i$
            $\mathcal{L} \leftarrow f(x, \theta_i^{(t)})$

            $\theta_i^{(t)} \leftarrow \texttt{InnerOpt}(\theta_i^{(t)}, \nabla\mathcal{L})$
        end for
    end for
    $\Delta^{(t)} \leftarrow \frac{1}{k}\sum_{i=1}^{k}(\theta^{(t-1)} - \theta_i^{(t)})$
    $\theta^{(t)} \leftarrow \texttt{OuterOpt}(\theta^{(t-1)}, \Delta^{(t)})$
end for

FSDP: Intra-Node Opt.

- Each node consists of several GPUs
  - E.g., 8xH100s
- Inside each node, training is done by PyTorch's FSDP2
- Uses ZeRO 3, shards the following:
  - Model weights
  - Gradients
  - Optimizer states

Elastic Device Mesh



Training Structure

Int8 Quantization

- DiLoCo is not efficient enough
- Idea: train with fp32, communicate with int8 --> 4x size
- Quantize the pseudo-gradients, not weights during training --> much more stable

Fault Tolerance

- Handling dynamic node participation is a major challenge:
  - **Challenge #1:** allow new nodes to join an ongoing training session without disrupting active nodes
  - **Challenge #2:** maintaining training continuity when nodes fail or leave the training run

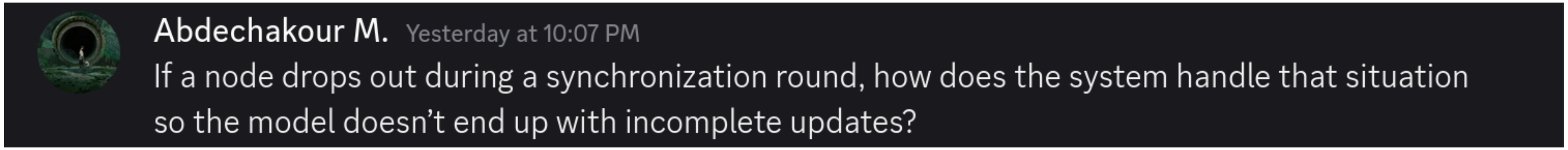
Peer to Peer Checkpoint Transmission

- Where does a new node get the current model weights?
- Authors looked at two methods:
  - **Non-blocking sync:** directly downloads the checkpoint from any available active peer while training continues
  - **Blocking sync:** Active nodes pause training while the new peer downloads the checkpoint directly from one of the active nodes
- Blocking sync was chosen due to simplicity & stability
  - New nodes join once every few days

Vinisha 12:04 AM
PRIME can sync checkpoints in two ways: blocking (which pauses training) and non-blocking (which keeps training going).
Why was the non-blocking approach selected as more practical for the INTELLECT-1 training run, even though it stopped active training?

Node Removal and Failures

- Nodes leave the training process for two reasons:
  - **Planned Exit:** gracefully exiting nodes
  - **Crash:** unexpected hardware or software error
- Solution --> heartbeat mechanism:
  - Send a message every 2 seconds
  - Node considered dead if no message after 6 seconds

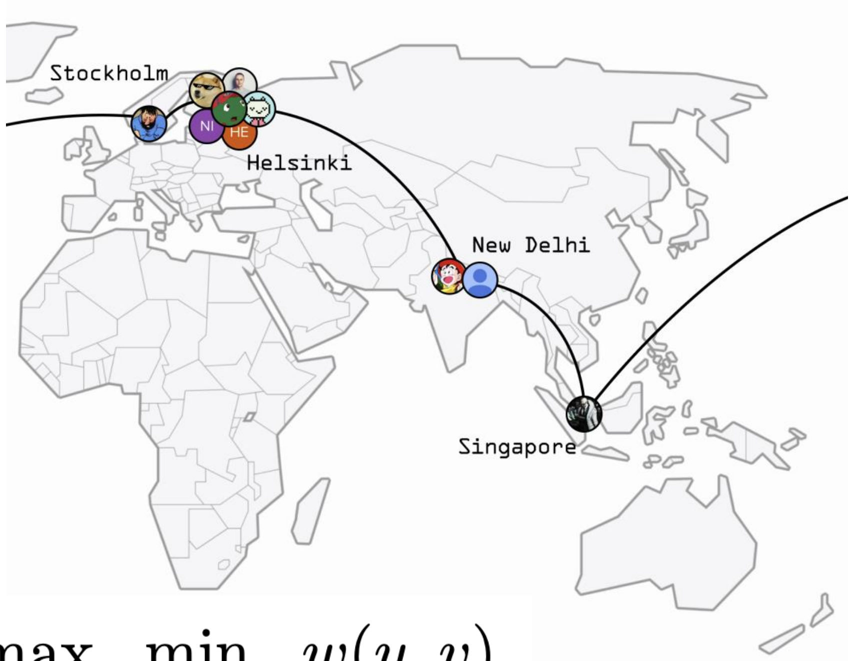


Networking

- All participating nodes were connected through a VPN:
  - **Reason #1:** Security
  - **Reason #2:** PRiME uses gloo library, which requires VPN connected nodes
  - **Reason #3:** Public IP routing leads to poor/variable bandwidth. A VPN optimizes p2p connections between nodes, due to modifying the routing of packets

Networking: All-Reduce

- Inter-node communication is done is a ring all-reduce
- How is the ring constructed:
  - Nodes continually measure inter-node bandwidth
  - An optimal ring-order of nodes is then constructed
  - Max-min Hamiltonian cycle (variation of TSP): $\max_{C \in \mathcal{C}} \min_{(u,v) \in C} w(u, v),$



INTELLECT-1 Training

Architecture - Setup

- 30 total nodes
- Max 14 concurrent nodes
- 8 data-centers
- 3 continents
- Node: 8xH100 GPU
- H: 100
- Pseudo-gradients: int8

Obadaalbaba Yesterday at 3:06 PM
How does Intellect-1 mitigate the impact of straggler nodes during federated synchronization, and does the system employ any form of adaptive weighting to avoid penalizing faster clients?

LLM - Architecture

- Llama 3
- Pre-training:
  - 42 days
- Post-training:
  - 16 SFT steps
  - 8 DPO steps
  - Merge 16 candidate models

- Fine Tuning Datasets:
  - 3 new released
  - 3 instruction following
  - 3 domain specific
  - 4 tulu-3 persona Datasets

Compute Efficiency

MFU: Model FLOPs Utilization [1]

$$\frac{\text{Actual FLOPs per second}}{\text{Theoretical FLOPs per second}}$$

Scenario	MFU (%)	Inner step time, min	Median All-Reduce time, s	Compute Util (%)
Baseline (no comm)	43.3	38	-	100
USA	41.4	38	103	95.7
USA + Europe	37.1	38	382	85.6
Global	36.0	38	469	83.0

Table 2: Performance metrics for training across different geographical configurations. Compute utilization refers to the proportion of time the training is not communicating with other nodes.

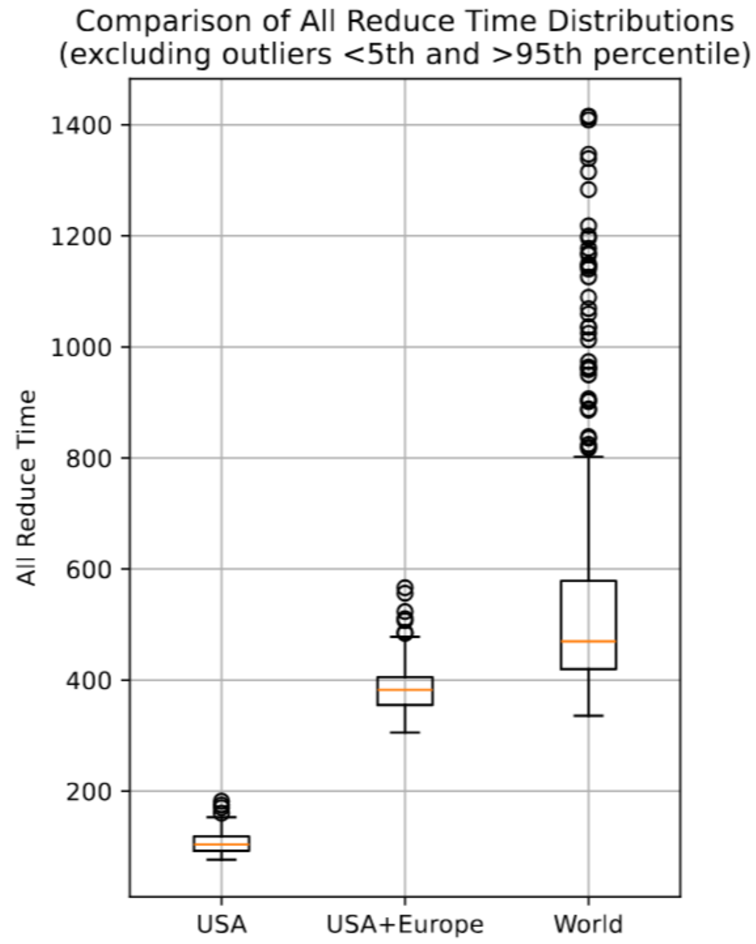
Geographically spread = longer All-Reduce.   Limited impact on compute utilization.

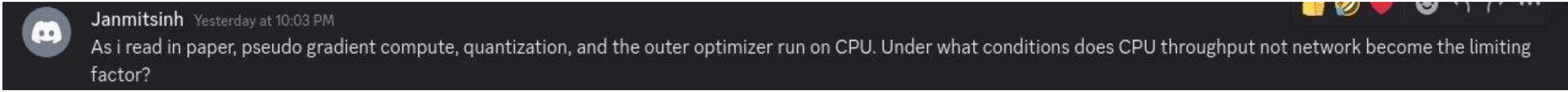
[1] : https://arxiv.org/pdf/2204.02311

Network Efficiency

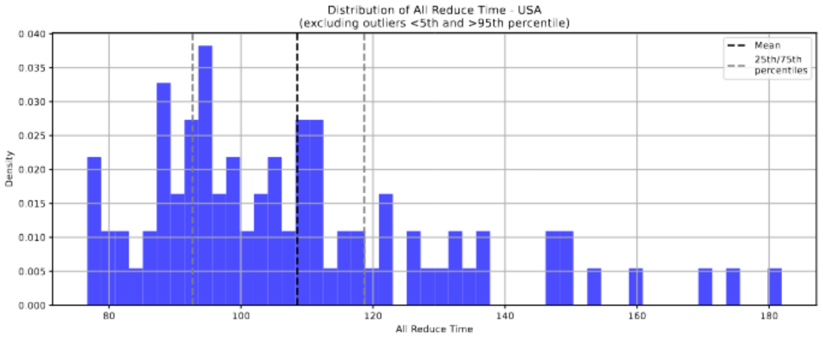
- All-Reduce time highly impacted
  - ~ Times 4 from USA to Global

- Comparison:
  - Save on disk: ~ 1 minute
  - Overhead from CPU based operations: 5-10 seconds

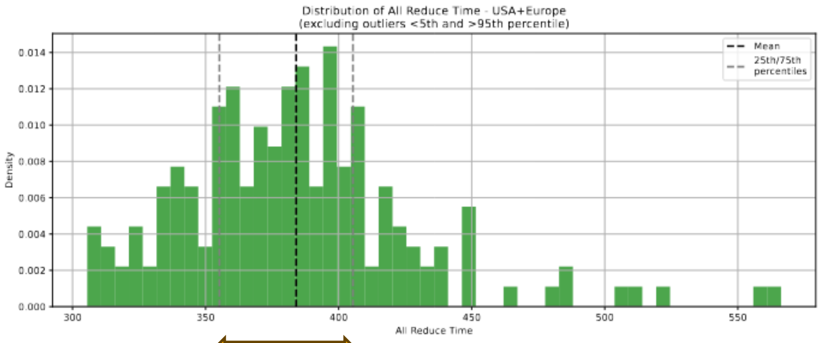




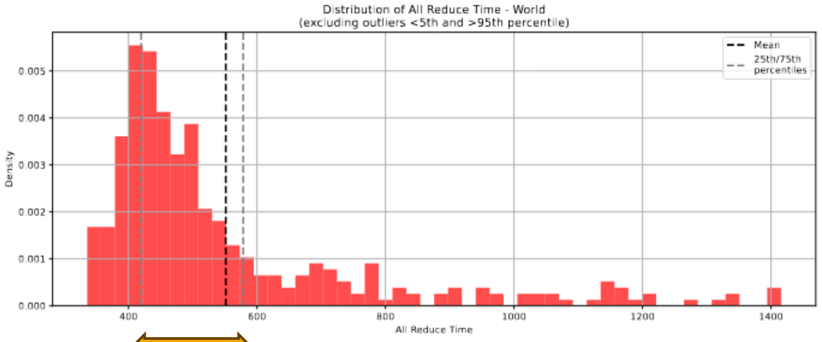
Geographical impact



~30s

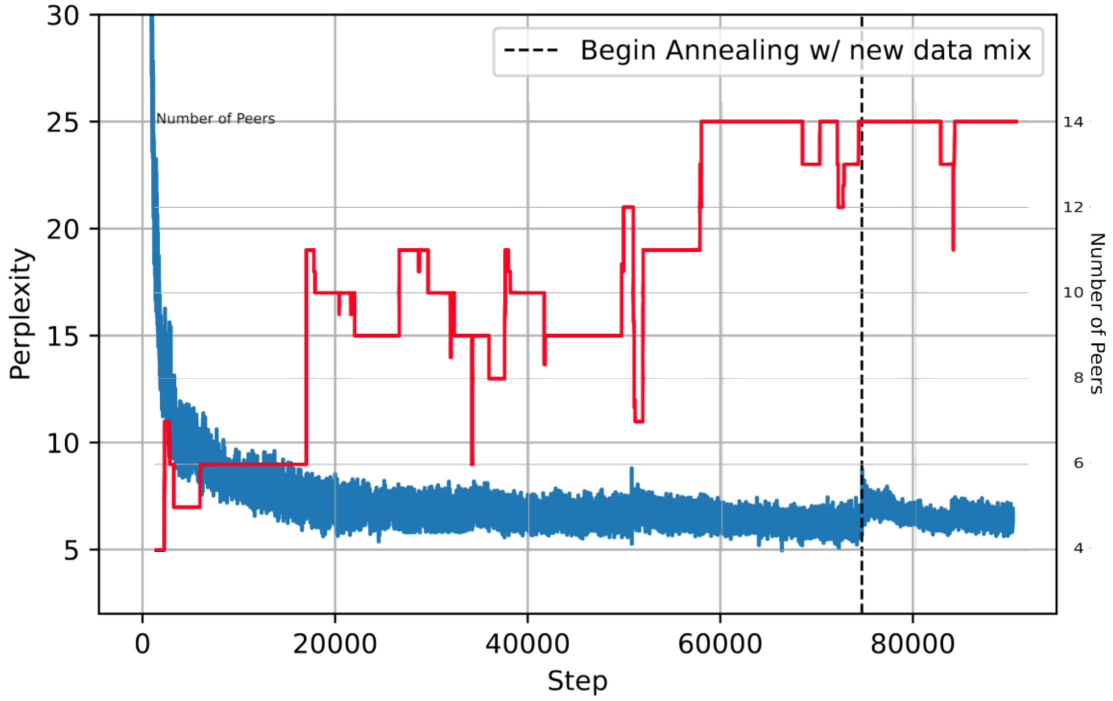


~50s



~180s

Resilience to Node Changes



- Training continues to improve with nodes gradual addition/deletion
- Instability when many nodes leave (4/12)

Model Performance

Pre-Training

Model	Size	Tokens	MMLU	HellaSwag	ARC-C
INTELLECT-1	10B	1T	37.5	72.26	52.13
MPT-7B (Team, 2023)	7B	1T	26.8	77.41	46.67
Falcon-7B (Almazrouei et al., 2023)	7B	1.5T	26.2	78.23	47.61
Pythia-12B (Biderman et al., 2023)	12B	300B	26.5	68.83	40.61
LLM360-Amber (Liu et al., 2023)	7B	1.3T	24.5	74.08	42.75
LLaMA-7B (Touvron et al., 2023a)	7B	1T	35.1	78.19	50.43
LLaMA-13B (Touvron et al., 2023a)	13B	1T	46.9	81.05	56.14
LLaMA2-7B (Touvron et al., 2023b)	7B	2T	45.3	78.64	54.10
LLaMA2-13B (Touvron et al., 2023b)	13B	2T	54.8	82.58	59.81
	GPQA	**GSM8K**	**TruthfulQA**	**Winogrande**	**BBH**
INTELLECT-1	26.12	8.1	35.47	65.82	32.97
MPT-7B	25.67	8.3	33.43	71.11	32.88
Falcon-7B	23.66	4.9	34.28	70.32	33.00
Pythia-12B	24.33	4.09	31.83	65.27	31.66
LLM360-Amber	27.01	4.32	40.80	65.35	31.95
LLaMA-7B	23.21	9.7	34.33	72.06	32.86
LLaMA-13B	26.34	17.3	39.48	76.16	39.74
LLaMA2-7B	25.89	13.5	38.75	74.03	34.46
LLaMA2-13B	25.67	24.3	37.38	77.35	41.68

Table 3: Base model evaluation results across various benchmarks, measured against similarly sized open-source models pre-trained in a centralized setting on comparable amounts of total tokens.

Post-Training

Model	Size	Tokens	MMLU	HellaSwag	ARC-C
INTELLECT-1-INSTRUCT	10B	1T	49.89	71.42	54.52
MPT-7B-Chat	7B	1T	36.29	75.88	51.02
Falcon-7B-instruct	7B	1.5T	25.21	70.61	45.82
LLM360 AmberChat	7B	1.4T	36.02	73.94	43.94
LLaMA2-7B-chat	7B	2T	47.20	78.69	53.33
LLaMA2-13B-chat	13B	2T	53.51	82.47	59.73
	GPQA	**GSM8K**	**TruthfulQA**	**BBH**	**IFEval**
INTELLECT-1-INSTRUCT	28.32	38.58	42.16	34.85	40.39
MPT-7B-Chat	26.79	8.26	35.22	32.30	14.39
Falcon-7B-instruct	26.34	4.93	44.13	31.98	24.82
LLM360 AmberChat	27.23	6.14	40.80	31.14	18.71
LLaMA2-7B-chat	28.57	23.96	45.58	35.50	45.80
LLaMA2-13B-chat	28.35	37.15	44.12	39.05	46.88

Table 4: Post-trained model evaluation results across various benchmarks.

Conclusion

Conclusion & Future Work

- 1st 10B LLM trained collaboratively across the world
- Features:
  - PRIME framework + ElasticDeviceMesh
  - DiLoCo-FSDP2 hybrid
  - Robust
- Efficiency:
  - 400x to 2000x communication BW reduction vs DP
  - High GPU utilization

- Future Work:
  - PCCL ("Prime Collective Communications Library"): To adapt to global internet limitations
  - Find incentives for community to contribute (à la BitCoin)
  - Train bigger models (now Intellect-2, 32B)

Limitations

- Not heterogenous
  - 8x H100 per node
  - CPU + memory
- 10 minutes global sync
  - Would Ring all-reduce scale?
- Stability when ~30% nodes lost
- Not compared to Llama 3
- Choice of Async vs Blocking
- Issues with VPN stability

Discord questions

Arpnik Yesterday at 8:58 PM

How can the training framework detect providers that submit adversarial gradients or subtly biased updates that don't look like random noise but still degrade convergence, especially when updates are sharded and aggregated through ring-all-reduce?

Quiz Questions

Question 1

- What methods do the authors use to decrease the amount of communication between nodes?

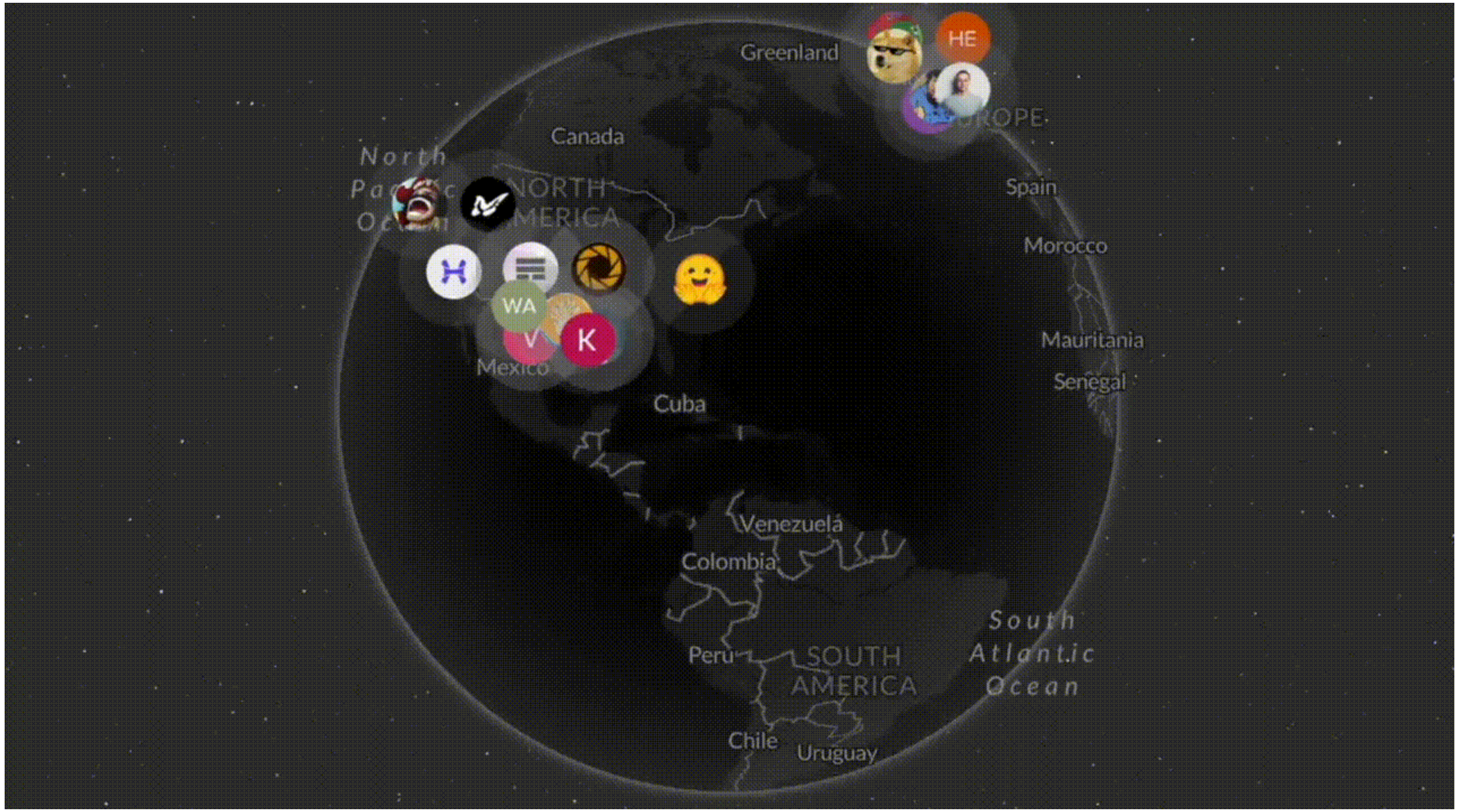
Question 2

- How does the system detect and handle node failures?

Question 3

- Why does global training require continuous bandwidth measurement?

The End


Greenland
HE
EUROPE
Canada
North Pacific Ocean
NORTH AMERICA
Spain
Morocco
WA
K
Mexico
Mauritania
Senegal
Cuba
Venezuela
Colombia
South Atlantic Ocean
Peru
SOUTH AMERICA
Chile
Uruguay