Hierarchical Distributed Low-Communication (HeDiLoCo) Training

Final Report

Romeo Dean
Harvard John A. Paulson School of Engineering and Applied Sciences
Cambridge, MA, USA

Giovanni M. D'Antonio
Harvard John A. Paulson School of Engineering and Applied Sciences
Cambridge, MA, USA

ABSTRACT

Frontier AI models are growing to multiple trillions of parameters, pushing companies to invest billions into new datacenter construction and accelerator purchases. As AI companies strive to keep up with current scaling trends [6], they are expected to push well past the capacity of any single datacenter [5]. This trend forces developers to consider multi-datacenter training strategies. However, geographically distributed datacenters introduce high-latency, low-bandwidth interconnects that make traditional synchronous training—requiring global parameter aggregation at every step—prohibitively expensive.

We present **Hierarchical Distributed Low-Communication (HeDiLoCo)** training, a framework inspired by DiLoCo [1], that extends asynchronous training with flexibility for hierarchical topologies. Workers in the same campus can synchronize frequently at low latency, while synchronization across distant regions can occur less frequently, to efficiently balance communication overhead with model convergence speed.

Our small experiments with a 100K parameter Transformer on the tiny Shakespeare dataset demonstrate that HeDiLoCo can **speedup training times relative to a synchronous baseline by 100 times while only incurring a 1-2% final validation loss penalty**. Our results highlight HeDiLoCo's promise as a flexible, scalable, cost-effective framework for future multi-trillion parameter, multi-datacenter AI training.

1 INTRODUCTION

Frontier Artificial Intelligence (AI) models now contain trillions of parameters, leading to training costs that can exceed $1 billion. These exponential scaling trends are driven by intense AI company competition and the rapid growth of model capabilities. However, traditional compute colocation strategies face bottlenecks, such as limited power availability, construction timelines, and the inability to build new datacenters fast enough to meet demand. For example, Microsoft recently announced a $7 billion investment in fiber cabling to interconnect its AI-dedicated infrastructure across regions [7].

A natural solution to these challenges is distributed training across geographically dispersed data centers. Unfortunately, this approach introduces formidable issues: high-latency and bandwidth-limited interconnects make traditional synchronous training—requiring global parameter aggregation at every step—prohibitively inefficient. To address these limitations, we propose **Hierarchical Distributed Low-Communication (HeDiLoCo)** training, a novel framework that reduces communication overhead by leveraging hierarchical synchronization structures tailored to real-world data center topologies.

We propose **Hierarchical Distributed Low Communication (HeDiLoCo)** training, an approach that extends the asynchronous concept introduced by DiLoCo [1] with a hierarchical structure. While DiLoCo reduces communication by synchronizing every H steps, HeDiLoCo further differentiates between intra-campus (lower latency) and inter-campus (higher latency) connections. Workers within the same region synchronize frequently, while workers across distant regions synchronize less frequently. This hierarchical approach can greatly reduce communication costs while maintaining robust model convergence, reflecting real-world landscapes where datacenters cluster into campuses and states.

1.1 Problem Definition

The exponential scaling trends in Artificial Intelligence (AI) models, driven by fierce competition among AI companies, have resulted in training costs surpassing $1 billion and compute demands that exceed the capacity of any single datacenter. Traditional colocation strategies face significant bottlenecks, including power limitations, construction delays, and the logistical challenges of building new facilities quickly enough to meet demand. Distributed training offers a natural solution but introduces new challenges.
The key challenges we address include:

(1) **Minimizing communication overhead**: High-latency, bandwidth-limited links between geographically distributed data centers make global synchronization

expensive and inefficient. Reducing the frequency of these global updates is critical to improving training scalability.
(2) **Maintaining model convergence**: Asynchronous updates inherently risk model divergence. A careful balance between local and global synchronization intervals is essential to achieve robust model performance.
(3) **Scalability and flexibility**: The solution must efficiently adapt to larger models, more complex data center topologies, and heterogeneous infrastructure, such as mixed hardware capabilities.

Novelty and Relevance: HeDiLoCo introduces a hierarchical synchronization strategy that differentiates between intra-campus (low latency) and inter-campus (high latency) connections, reflecting real-world data center topologies. By reducing the frequency of costly global synchronizations, HeDiLoCo offers significant financial and time savings. For example, in AWS-based deployments across us-west-2 and us-east-1 regions, HeDiLoCo demonstrated promising improvements in communication efficiency while maintaining strong convergence.

2 RELATED WORK

Our work builds upon Google's **DiLoCo** (Distributed Low-Communication) framework [1], which trains language models asynchronously by synchronizing parameters every H steps. DiLoCo successfully demonstrated the viability of asynchronous training across geographically distributed data centers with 60M–400M parameter Transformer models, achieving strong convergence on the C4 dataset. However, DiLoCo treats all workers uniformly, ignoring the hierarchical topologies commonly found in real-world data center deployments.

Limitations of DiLoCo: While DiLoCo reduces communication by synchronizing infrequently, it lacks flexibility to optimize for layered hierarchical structures, such as those found in multi-campus or multi-region networks. This oversight limits its efficiency and scalability when applied to more complex topologies.

Beyond DiLoCo, other approaches in distributed training include:

- **Federated Learning Methods**: Techniques such as Federated Averaging [3] focus on aggregating model updates from distributed clients. While these methods prioritize privacy and client autonomy, they do not explicitly address latency and bandwidth constraints in hierarchical data center settings.
- **Parameter Server Architectures**: Distributed frameworks like MapReduce-based parameter servers [2] enable large-scale model training, but their centralized synchronization paradigm introduces bottlenecks in high-latency scenarios.
- **Asynchronous SGD Methods**: Approaches like Hogwild [4] reduce synchronization frequency but lack the awareness of hierarchical topologies, making them less suitable for geographically distributed environments.

HeDiLoCo's Novelty: Our framework introduces a hierarchical synchronization structure tailored to real-world data center topologies, where intra-campus synchronization occurs frequently, and inter-campus synchronization occurs less often. By leveraging this hierarchical awareness, HeDiLoCo fills a critical gap left by prior methods, enabling cost-effective and scalable distributed training for next-generation AI models.

3 DESIGN AND TOPOLOGY

3.1 High-Level Approach

HeDiLoCo's design introduces multiple layers of synchronization:

- **Local Synchronization (H_1 steps)**: Workers within the same campus (low-latency connections) synchronize frequently, every H_1 steps. For example, in our experiments, $H_1 = [2, 5, 10, 25]$ steps were used.
- **Global Synchronization (H_2 steps)**: Workers across geographically distant regions (high-latency connections) synchronize less frequently, every H_2 steps. For instance, $H_2 = [100, 200, 300]$ steps were chosen for experimentation.

This hierarchical synchronization strategy reflects the real-world organization of data centers, where campuses are clustered locally but separated by high-latency, bandwidth-limited wide-area network (WAN) links.

3.2 Network Topology and Deployment

Our initial evaluation used a 2x2 setup with four workers organized hierarchically:

- **Local workers**: Paired within the same AWS region, specifically either us-west-2 (Oregon) and us-east-1 (Northern Virginia).

The AWS deployment configuration included:

- Non-overlapping Virtual Private Clouds (VPCs) for each region.
- Configured peering connections between VPCs.
- Custom route table rules and internet gateways for encrypted weight communication.
- Security group rules to ensure secure data transfer.

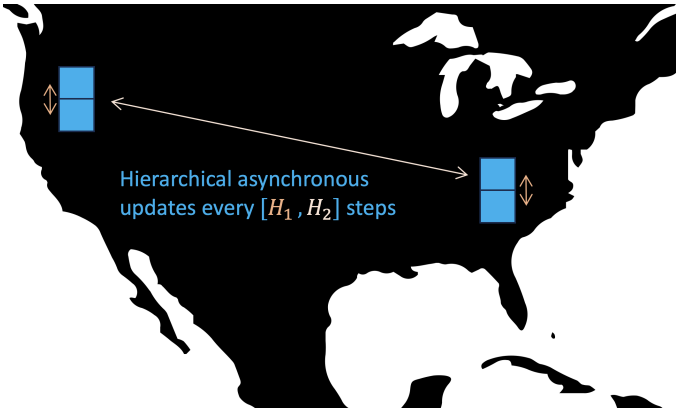

Figure 1: Illustration of 2x2 HeDiLoCo

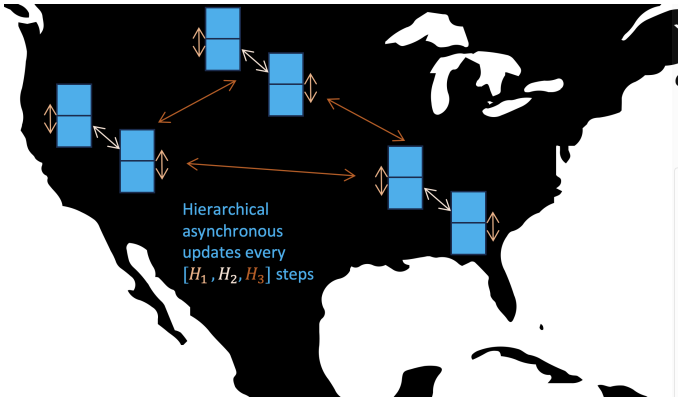

Figure 2: Illustration of 2x2x3 HeDiLoCo

3.3 Latency Observations

Networking latency was a key consideration:

- **Local all-reduce latency**: Approximately 80ms for intra-campus synchronization.
- **Regional all-reduce latency**: Approximately 13 seconds for inter-campus synchronization.

These observations highlight the stark differences in communication overhead between local and regional synchronizations, motivating the hierarchical approach.

3.4 Scalability and Model Size

We scaled the model size from 100K to 600K parameters, observing:

- An increase in regional all-reduce latency from 12.7 seconds to 18.3 seconds.
- A corresponding increase in local all-reduce latency from 80ms to 463ms.

3.5 Flexibility

While the current 2x2 configuration demonstrates the viability of hierarchical synchronization, HeDiLoCo is designed to scale to more complex topologies, such as 2x2x3. Future enhancements will include adaptive synchronization intervals and support for heterogeneous worker capabilities to further optimize performance across diverse deployments.

4 IMPLEMENTATION

Our implementation of HeDiLoCo demonstrates its feasibility in both simulated and real-world environments, leveraging a hierarchical synchronization structure to minimize communication overhead.

4.1 Training Setup

We used a 100K-parameter Transformer model trained on the Shakespeare dataset (1MB of text) to quickly test the viability of HeDiLoCo. This dataset size allowed for rapid experimentation and initial observations. The training setup was deployed across AWS regions.

4.2 Synchronization Frequencies

- **Local Synchronization ($H_1 = [2, 5, 10, 25]$ steps)**: Workers within the same AWS region synchronized their parameters frequently, using lower-latency connections to reduce model divergence.
- **Regional Synchronization ($H_2 = [100, 200, 300]$ steps)**: Workers across regions synchronized less frequently, leveraging a modified Federated Averaging method with Nesterov momentum to accelerate convergence and enhance stability.

4.3 Hyperparameters and Scaling

Experiments were conducted with $H_1 = [2, 5, 10, 25]$ steps and $H_2 = [100, 200, 300]$ steps. Holding the model size constant at 100K parameters.

4.4 Security and Communication

To ensure the security of distributed training:

- Encrypted weight communication was implemented across regions using secure internet gateways.
- Security group rules were configured to allow only authorized data transfers between workers.

This robust implementation framework not only validates HeDiLoCo's hierarchical synchronization but also ensures its practicality in real-world, geographically distributed training environments.

5 EVALUATION

We evaluate HeDiLoCo by comparing it to a traditional synchronous baseline, studying its scalability, latency robustness, and performance regarding model convergence and communication costs. These experiments build upon insights from both our midterm results and expanded real-world tests.

5.1 Experimental Setup

Hardware and Environment: The evaluation was conducted on AWS infrastructure:

- **Instances Used**: Two c5.4xlarge instances in us-west-2 (Oregon) and two c5.4xlarge instances in us-east-1 (Northern Virginia).
- **Network Configuration**: Non-overlapping Virtual Private Clouds (VPCs) were configured with secure peering connections, custom route table rules, and internet gateways to enable encrypted weight communication.
- **Networking Latencies**: Local synchronizations (within the same region) exhibited a latency of 80ms, while global synchronizations (across regions) showed latencies of 13 seconds.

Metrics: We used the following metrics to assess performance:

- **Training Loss**: Tracks model learning progress across steps.
- **Validation Loss**: Measures generalization on unseen data.

5.2 Results and Findings

HeDiLoCo closely tracked the synchronous baseline for training loss in most configurations stabilizing around 1.75 after 10,000 steps. Figures 2 and **??** illustrate that, despite less frequent global synchronizations ($H_2 = 250$ steps), the hierarchical approach maintained strong convergence with only a slight penalty in validation loss.

5.2.1 Overall picture. Figure 3 shows the comprehensive training and validation loss curves for the synchronization frequencies we attempted and compares HeDiLoCo with a Synchronous baseline. We see that HeDiLoCo closely tracks synchronous convergence in most cases, and seems to actually over-fit in one of the cases where it was synchronizing more frequently. The synchronous baselines demonstrate the loss curves that a theoretical training run synchronizing at every step would achieve (but is run in a co-located setting to avoid hour-long runs).

5.2.2 Specific Hierarchical run. In this section we dive into more detail on a specific run where we set $H_1 = 5$ steps, $H_2 = 200$. Figure 4 shows the training loss curves and figure 5 shows various details about the synchronization times and utilization, and also shows the weights updates being sent.

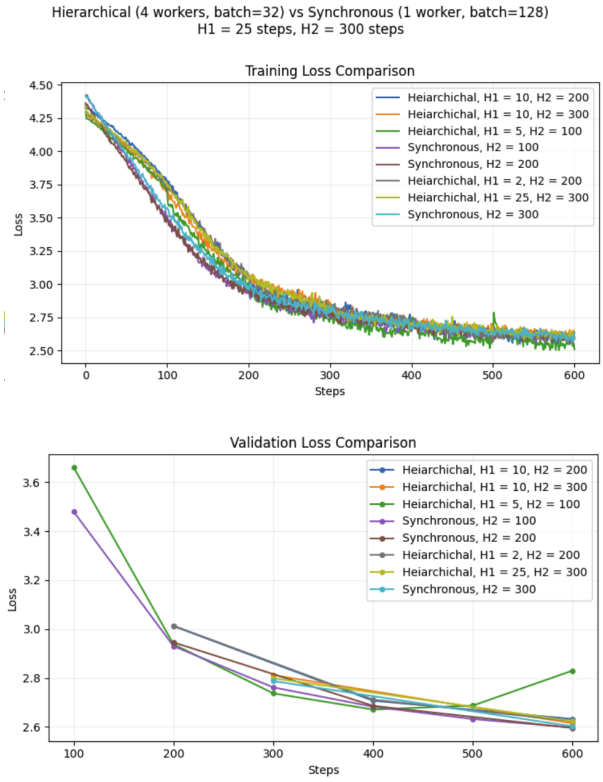

Figure 3: Training Loss and Validation Loss Over 600 Steps: Hierarchical (4 workers, batch=32) vs. Synchronous (1 worker, batch=128) in various combinations of H_1 and H_2 synchronization frequencies.

5.2.3 Synchronous baselines. A fair synchronous comparison to our HeDiLoCo experiments would run in the same distributed setting but with a global all reduce at every training step. We did not implement this baseline, but measured that it would take 2 hours to run given the 13 second all reduce time per step. Instead, we pretended the compute was co-located, and trained with 4x larger batch size and learning rate to see how well a synchronous run would do on the dataset. Figure 6 shows this relationship between the large synchronous training time and its estimated validation loss using this proxy.

5.2.4 Model scaling test. As an additional exercise we implemented a 600K parameter transformer and ran it in the same configuration with $H_1 = 5$, $H_2 = 200$. We found as shown in figure 7 that the model saw much worse convergence compared to a synchronous method, which may be a sign that

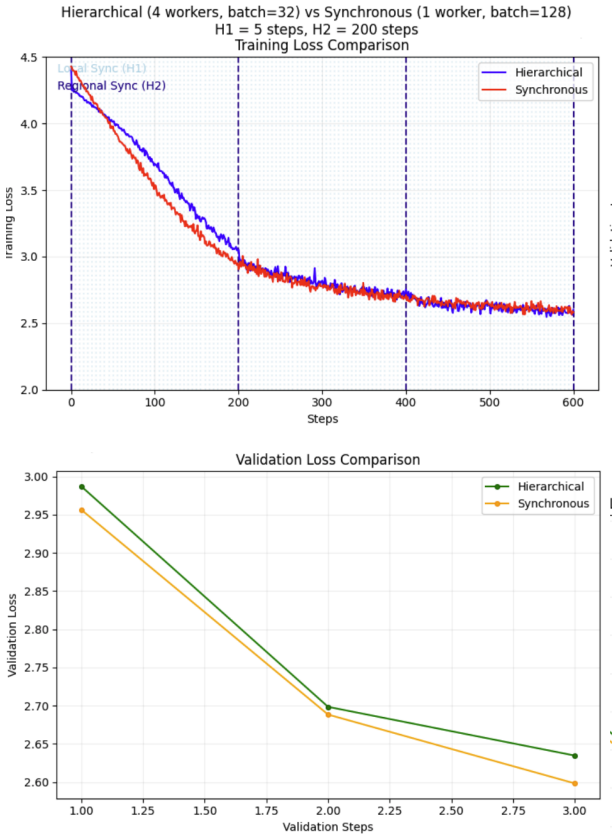

Figure 4: Training Loss and Validation Loss Over 600 Steps: Hierarchical (4 workers, batch=32) vs. Synchronous (1 worker, batch=128) with $H_1 = 5$ and $H_2 = 200$ synchronization frequencies.

scaling model size puts an additional burden on asynchronous algorithms, but is more likely just a function of less exploration in hyper parameter space. The local and global synchronization latencies only grew marginally, especially in the glocal sync step, despite the model size growing to 600K, which shows communication is mainly latency and not bandwidth bottlenecked, as expected.

- Local Sync Time: Increased from 80ms to 450ms
- Global Sync Time: Increased from 13 seconds to 18 seconds

5.3 Overall Advantages and Scaling

Our results in this limited experimental setting show promising advantages for hierarchical distributed training with large communication overhead reduction at a low convergence penalty. The results showed:

- In comparison to the a fully synchronous distributed training run, HeDiLoCo can reduce training time up to 100x in our setting.
- Despite these communication overhead gains, we see a very small validation loss penalty of 1-2% on these training runs.

5.4 Latency Scenarios and Flexibility

HeDiLoCo demonstrates adaptability to diverse network conditions:

- A wide range of H_1 and H_2 values tested achieve significant training time gains and low convergence penalty.
- This flexibility suggests that HeDiLoCo should be well-suited for real-world multi-datacenter setups of various kinds, and might even be scaled down to within a datacenter where servers on opposite sides of a large cluster might form a lower layer of the hierarchy.

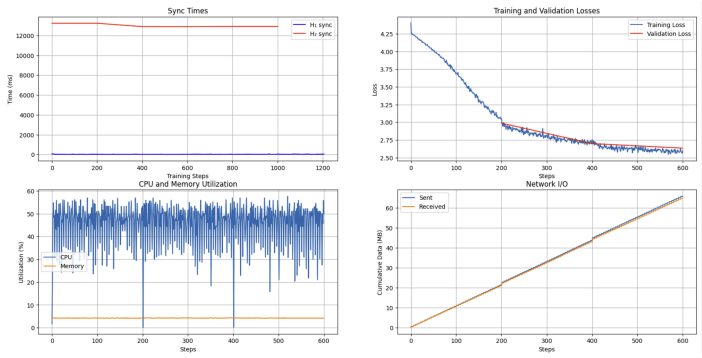

Figure 5: (Top Left) Local and Regional synchronization times (including encryption/decryption and all-reduce). (Top Right) training loss and validation loss. (Bottom Left) CPU and memory utilization curves (notice the downward spikes at global sync steps). (Bottom right) cumulative bytes sent on the network, reflecting the weights updates. $H_1 = 5$ steps, $H_2 = 200$ steps.

By focusing on training time and convergence performance metrics, our results show the potential practical benefits of using HeDiLoCo for training frontier AI models efficiently across geographically distributed data centers.

6 LIMITATIONS AND FUTURE DIRECTIONS

The results in this paper are promising for the HeDiLoCo framework but are also still very much just a basic 'proof of concept' implementation. Several limitations and

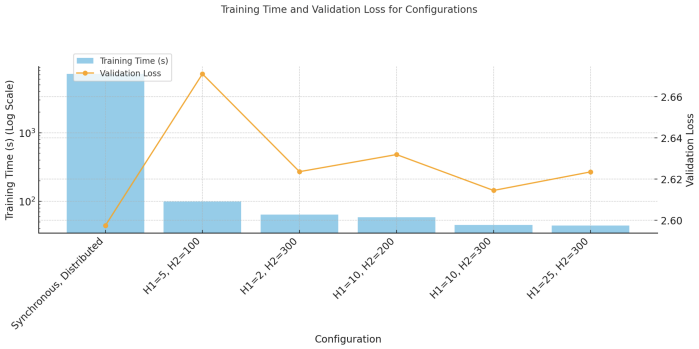

Figure 6: Validation loss and training time comparison for different configurations. These illustrate that the trade-off in convergence penalty for achieving a significantly lower training time are small. Notice the training time axis is on a log scale.

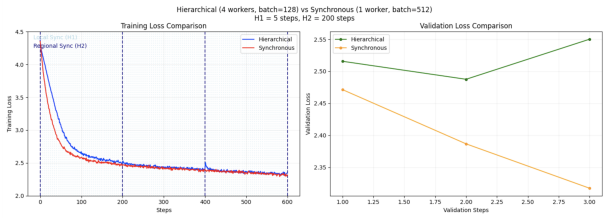

Figure 7: 600K parameter transformer scale-up. Training Loss and Validation Loss Over 600 Steps: Hierarchical (4 workers, batch=32) vs. Synchronous (1 worker, batch=128) with $H_1 = 5$ and $H_2 = 200$ synchronization frequencies.

(1) **Scaling to Larger Models, Datasets, and Training runs**: While our current experiments focused on 100K parameter models, future efforts would need to scale HeDiLoCo to models with millions or perhaps even billions of parameters. This would achieve the important goal of showing whether asynchronous training runs can be actually implemented in frontier development efforts.
(2) **Convergence Assurances**: The experiments we ran did not ensure that the synchronized baselines were representative of the best-case convergence possible with a synchronized training run, and we would need to be expand the training runs tested to show convergence being achieved by HeDiLoCo that is competitive with a convergence we know to be optimal.
(3) **Enhanced Hierarchical Structures**: Implementing more complex hierarchical topologies, such as 2x2x3 or beyond, will provide insights into the scalability of HeDiLoCo in increasingly large and complex networks and topologies.
(4) **Heterogeneous Worker Flexibility**: Real-world data centers often involve heterogeneous hardware (e.g., GPUs with varying capabilities). Future work should focus on adapting HeDiLoCo to optimize synchronization and communication frequencies dynamically and ideally automatically across such heterogeneous setups.
(5) **Hyperparameter Tuning and Dynamic Adjustments**: Fine-tuning synchronization intervals (H_1, H_2) based on network conditions, model size, and training progress is critical for further narrowing the convergence gap. Additionally, exploring advanced optimization techniques (e.g., adaptive learning rates and momentum strategies) to enhance stability. These could be made dynamic to adapt and correct course even during a training run.
(6) **Security and Network Enhancements**: It would be nice to show the penalty introduced by the encryption of the weights transfer and how this scales with larger models.

By addressing these challenges, HeDiLoCo has the potential to become a robust and scalable framework for training frontier AI models efficiently across geographically distributed environments.

7 CONCLUSION

Our final evaluations demonstrate that **HeDiLoCo's hierarchical approach** significantly reduces communication overhead while maintaining strong model convergence in highly geographically distributed training environments. By performing frequent local synchronizations and infrequent global synchronizations, HeDiLoCo effectively balances latency, bandwidth constraints, and model performance.
Key findings include:

- **Efficiency Gains**: HeDiLoCo reduced total training time by a factor of 100 times compared to a synchronous baseline for a 100K-parameter model. Regional synchronization latencies (13 seconds) and local synchronization latencies (80ms) illustrate its ability to optimize communication costs.
- **Convergence Penalty**: We were able to find synchronization configurations that only penalized final validation loss on our dataset by 1-2%.
- **Adaptability**: Various different synchronization frequencies were found to be robust in achieving this performance which shows the inherent promise of HeDiLoCo to be adaptable to many different real-world topologies.

Future Directions: Building on these results, future efforts should focus on scaling experiments to larger models and datasets, ensuring best-case-scenario convergence comparisons, improving adaptability for heterogeneous workers, and building in dynamic and automated adaptability for hyperparameters, optimizers and heterogeneous settings. Comparisons to other distributed learning baselines and implementation of advanced hierarchical structures (e.g., 2x2x3 setups) might further provide compelling evidence for the promise of HeDiLoCo.

As AI models grow in scale and complexity, surpassing the physical limits of single datacenter clusters, HeDiLoCo offers a scalable and efficient framework for distributed training. Its promise lies in bridging the gap between drastically reduced communication overhead while trading off very little in terms of model performance, providing a blueprint for practical, distributed AI development.

REFERENCES

[1] Arthur Douillard, Qixuan Feng, Andrei A. Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. 2024. DiLoCo: Distributed Low-Communication Training of Language Models. *arXiv preprint library* (2024). https://doi.org/10.48550/arXiv.2311.08105

[2] Mu Li, Li Zhou, Zichao Yang, Aaron Q. Li, Fei Xia, David G. Andersen, and Alex Smola. 2013. Parameter Server for Distributed Machine Learning. (2013). https://api.semanticscholar.org/CorpusID:2902150

[3] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR* abs/1602.05629 (2016). arXiv:1602.05629 http://arxiv.org/abs/1602.05629

[4] Augustus Odena. 2016. Faster Asynchronous SGD. (2016). arXiv:1601.04033 [stat.ML] https://arxiv.org/abs/1601.04033

[5] Dylan Patel, Danie Nishball, and Jeremie Eliahou Ontiveros. 2024. Multi-datacenter training: OpenAI's ambitious plan to beat Google's infrastructure. SemiAnalysis. https://semianalysis.com/2024/09/04/multi-datacenter-training-openai/ Accessed: 2024-12-11.

[6] Jaime Sevilla and Enrique Roldán. 2024. Training compute of Frontier AI models grows by 4-5x per year. https://epoch.ai/blog/training-compute-of-frontier-ai-models-grows-by-4-5x-per-year Online Resource.

[7] Lumen Tehnologies. 2024. Lumen - AI demand drives $5 billion in new business and massive expansion of the internet. https://ir.lumen.com/news/news-details/2024/AI-Demand-Drives-5-Billion-in-New-Business-and-Massive-Expansion-of-the-Internet/default.aspx Accessed: 2024-12-11.