Vol. 16 (2026): Proceedings of the 2026 International Conference on AI and Technological Innovation Applications (AITIA 2026)
E-ISSN: 3105-0522 P-ISSN: 3105-0514
DOI: https://doi.org/10.70267/aitia.2026421426

A Survey: Lossless Model Accuracy Data Partitioning and Training Parallelism Strategies for Distributed Machine Learning

Jinhui Xia*

Silesian College of Intelligent Science and Engineering, Yanshan University, Qinhuangdao, China

**Corresponding author: Jinhui Xia.*

Abstract

In the digital era of big data and large-scale models, conventional stand-alone computing power and storage approach fails to meet the demands of large-scale machine learning models, therefore, distributed data parallelism has been the key supporting technology for breaking hardware bottleneck. However, current distributed training is facing numerous actual conflicts in the practical applications: data partitioning and resource scheduling, heterogeneous clusters and global synchronization. This passage elaborates on the practical dilemmas of distributed machine learning training, achieving thorough integration of data partitioning patterns, parallel operation architectures and precision stabilization strategies by three dimensions: problems, methods and findings. The study points out that existing technology manifests multiple limitations, such as inefficient training iteration, poor flexibility of static partitioning, and the difficulty in simultaneously maintaining accuracy invariance and convergence consistency. Subsequent research can develop with the direction of building Non-Independent and Identically Distributed (Non-IID) adaptive partitioning mechanisms, constructing efficient resource coordination platforms and correcting local gradients, which provides reliable references for establishing high-precision distributed machine learning systems.

Keywords

distributed machine learning, accuracy losslessness, non-independent and identically distributed, gradient compensation

1. Introduction

In the current artificial intelligence field, large-scale parameter training of deep learning models necessarily involves associated data such as forward-pass activations, backward-pass gradients and optimizer states, which means that the compute power and memory capacity of a single node cannot satisfy large-scale parameters, getting into a hardware bottleneck [1]. From the theoretically optimal solution, the development of a single node could certainly meet our needs, but as models continue to grow, standalone resources are bound to be deficient. Therefore, developing distributed machine learning is an essential and optimal route for handling massive parameters [2]. Among the various distributed architectures, compared with model parallelism and pipeline parallelism, data parallelism has become the most widely adopted training topology model with high

generality. However, traditional data parallelism is actually fragile, because it generates a "barrel effect"—faster nodes have to wait for the slowest node to finish, but this can be solved through data partitioning [3].

In the practical heterogeneous environments, the execution efficiency of data parallelism and the final model's convergence accuracy depend on the granularity of the underlying data partitioning mechanism and the robustness of dynamic scheduling. Optimization exploration for heterogeneous data parallelism has gradually converged toward three frontier dimensions centered on "dynamic partitioning and compensation": the data partitioning dimension, the data parallelism dimension and the mathematical precision assurance dimension. This paper presents a survey organized around these three dimensions. Chapter 2 introduces the relevant theoretical background; Chapter 3 reviews the cutting-edge literature along the main lines of "problem layer," "method layer," and "finding layer"; Chapters 4 and 5 offer a comprehensive discussion and outline directions for future research.

2. Theoretical Background

The core mechanism of distributed data parallelism can be summarized as: data partitioning, local computation, gradient aggregation and parameter update. Under this architecture, the global dataset is divided and allocated to individual worker nodes; each node computes local gradients based on its local data. And then, these local gradients are subsequently aggregated through a weighted combination to obtain the global gradient, which is used to update the global parameters before starting the next iteration.

The three key dimensions of distributed data parallelism: First, in the data partitioning dimension, static partitioning is not flexible enough. It is necessary to develop compute-aware data partitioning algorithms, which can break away from the rigidity of static uniform partitioning and dynamically allocate workloads according to the computing power of different nodes in a heterogeneous environment. Besides, in the data parallelism dimension, the focus is on exploring dynamic batch adjustment and real-time resource coordination mechanisms. Based on the real-time status of each node, training tasks are assigned and adjusted for dealing with the dynamic fluctuations of the cluster's physical state and improving data processing efficiency. Finally, in the mathematical precision assurance dimension, research centers on the local gradient imbalance problem that arises from non-uniform partitioning. Exploring accuracy-preserving gradient compensation and transformation strategies ensures that the underlying heterogeneous acceleration does not occur at the expense of the model's final convergence performance.

The convergence of distributed training is highly vulnerable to degradation, and the errors mainly come from three sources: firstly, data heterogeneity of nodes, which can cause local drift in the model update direction. Secondly, "stale gradients" resulting from communication delays. Thirdly, the gradient compression or truncation for reducing communication cost, introduces the numerical discretization noise. These underlying theories comprise the basis for analyzing the "accuracy-preserving" challenge in distributed training.

3. Literature Review

As research logic, this section systematically divides the existing literature into three layers: the problem layer, the method layer, and the finding layer.

3.1 Problem Layer: Core Challenges in Distributed Training

When scaling machine learning to large clusters, researchers found that simply adding more system resources does not get a linear improvement in performance. As revealed by Dean et al. [4], due to the inevitable long-tail latency in heterogeneous clusters, system throughput is often bottlenecked by the slowest node. Early naive random data partitioning, which ignored the actual distribution of data, could easily lead to computational load imbalance across nodes. With the deep research, data heterogeneity was identified as the significant problem that undermines training accuracy. When the class distribution of each node's data is extremely non-uniform, it causes a severe "client drift" phenomenon [5]. Meanwhile, without the ability to monitor each node's real-time performance and make timely adjustments, processing data from multiple nodes concurrently may lead to the decline of model efficiency and disorder, and the barrel effect occurs frequently [6]. This has the high possibility to trigger a secondary imbalance in local gradients, if without a targeted compensation approach, it can cause a dramatic deviation in the global gradient. Besides, Alistarh et al. [7]

propose that gradient compression originated from communication cost as another major pain point. This numerical noise that is introduced by discretization operations directly makes a threat to the convergence performance of the model in distributed frameworks that lack precision correction.

3.2 Method Layer: Prevailing Strategies for Data Partitioning and Parallelism

3.2.1 Parallel Scheduling Optimization for Heterogeneous Computing Power

To address the unavoidable long-tail latency in heterogeneous clusters, researchers have come up with solutions from two dimensions: slow-node govern and dynamic scheduling.

Slow-node govern: In heterogeneous environments, special treatment is applied to slow nodes. The core logic is that after the parameter server assigns tasks, by four metrics: per-step latency, GPU load, data loading time, and NCCL communication time, the parameter server determines whether a node is slow. According to the severity, taking local downsampling and adaptive micro-batch adjustment for node offline [8]. This approach is highly compatible, applying different measures to different slow nodes and ensuring the model's accuracy and resource utilization.

Dynamic scheduling: This method exerts an effect from four aspects: samples, per-card hyperparameters, resources and clusters, and communication. It assigns large-scale samples to high-performance nodes, reducing the situation of overload on low-compute-power nodes. It synchronizes the iteration of the entire group by dynamically reducing samples and adjusting gradients for slow nodes. Isolating Idle background cores, scaling down and releasing redundant computing resources, taking faulty nodes offline ensures the high overall model performance of the model. Finally, during gradient aggregation, high-performance network cards aggregate data within the rack and enable cross-machine global communication, while low-performance network cards are only responsible for aggregating data inside the rack [9]. This method dynamically schedules resources among nodes and dramatically enhances the resource utilization.

3.2.2 The Client Drift Problem Caused By Non-IID Data

To address the client drift problem caused by non-IID data, proximal regularization and variance reduction are the current mainstream solutions.

Proximal regularization: a hard constraint at the parameter level, restricting the final weights.

The local optimization object:$\min L_i(w)+\frac{\mu}{2}\|w-w_g\|_2^2$

w_g is the global model distributed in the current round, and$\mu$is the proximal penalty coefficient.

In the FedProx framework, in the local training, if the parameters deviate substantially from the global weights w_g, $\|w-w_g\|_2^2$ it will increase rapidly. This is an in-model mechanism that causes the penalty loss to grow quickly, which helps to restrict the local parameters from converging unboundedly towards the local private data optimum. It prevents the large drift at the weight level [10]. By restricting the parameter distance, this mechanism forces the model's tendency to drift under poor data partitioning, guaranteeing the global convergence.

Variance reduction: Correcting the directional bias and adjusting the descent direction at the gradient level.

Local gradient correction: $\tilde{g}=g_{local}-c_i+c$

c:global control term, c_i:client-private control term

Utilizing the idea of variance reduction, due to the Non-IID distribution of local data differs across clients, each client helding gradient bias is different. Therefore, we eliminate the local effect by subtracting the local inherent gradient bias $g_{local}-c_i$ [10], and then add c to pull the local gradient closer to the global gradient. This offsets the gradient bias in real time during local iterations, theoretically eliminating the client drift phenomenon.

3.2.3 Accuracy-Preserving Compression Strategies for Communication Bottlenecks

To break the "barrel effect" in concurrent training, the method is abandoning the random node assignment, emphasizing the task differentiation and supplemented by dynamic node elimination as a fallback optimization.

Task differentiation: Tasks are assigned based on the computing power and performance, resolving the barrel effect in synchronous training through artificial approaches. By scaling the workload, the training times of nodes are made to finish nearly synchronously, which helps that there are no nodes with a huge training time gap. This achieves mutual gains between system efficiency and model performance.

Dynamic elimination: Setting per-iteration computation time. If a node exceeds this time, it is forcibly taken offline, followed by a round cooldown to prevent jitter from transient spikes. After each round, a metric inspection is triggered. If a node stays within the threshold for two consecutive rounds, it is brought back online [11]. This method completely eliminates the synchronization waiting cost between fast and slow nodes, and with a mixed strategy of cooldown and metric-triggered re-connection to ensure the global data coverage.

3.2.4 Guaranteeing Model Accuracy for Communication Stress

Guaranteeing model accuracy while alleviating communication bottlenecks is the last line to address data heterogeneity and communication pressure in distributed computing.

Error feedback and memory mechanism: Although gradient compression dramatically reduces bandwidth pressure, a fraction of gradients necessarily misses out during the compression procedure. The core solution is to establish an error buffer pool on each local node, which stores the quantization error produced by each gradient compression and compensates it into the new gradient in the next iteration [12]. This compensation mechanism ensures that all minute gradient information eventually participates in the global update, enabling the system to achieve convergence accuracy identical to an uncompressed strategy, thus realizing the goal of accuracy-preserving distributed training.

3.3 Finding Layer: Achievements and Limitations of Existing Techniques

Research shows that the above methods not only have achieved remarkable results in specific scenarios, but also expose their limitations. For slow nodes, discarding the gradients of the slowest nodes represents a huge waste of computing resources. More critically, slow nodes often mean the specialized devices, directly discarding them means the model can never learn the unique data characteristics behind such devices, making a severe systemic bias.

Second, proximal regularization and variance reduction forcibly counteract data heterogeneity through mathematical constraints. Obtaining the theoretically optimal value for the penalty coefficientμis extremely difficult. Iμis set too large, the local model becomes overly correlated with the global model, leading to painfully slow convergence; if set too small, the client drift phenomenon is triggered frequently. In real dynamic clusters, the theoretically optimal μ is unattainable and can only be approximated, often at the cost of substantial computing power.

The limitation of task differentiation and node elimination lies in the fact that nodes with high computing power and high loss values are not eliminated, while those eliminated, even if later brought back online, can easily cause the system to fall into a continuously diverging heterogeneous load under large-scale iterations. Certain slow nodes that possess rare and valuable data may be permanently shut out by the scheduler due to high system latency, causing the model's generalization accuracy on that data to plummet off a cliff.

Error feedback indeed achieves accuracy preservation under extreme compression, but it transforms the communication crisis into a memory crisis. To store the quantization error, the algorithm requires allocating an error buffer pool locally with the exact same dimensions as the model gradient. This means that the system needs to establish an additional block of memory equal in size to the model parameters during backpropagation. In current large-model training, GPU memory is often the most scarce resource, which means that this way, which occupies massive memory usage is highly provoking breakdown in real deployment. Moreover, the buffer pool undergoes two vector additions or subtraction operations every round, and the accumulated computational cost stands out as a prominent limitation under huge parameter counts.

4. Discussion and Synthesis

A comprehensive look at the current state of distributed machine learning research reveals a common thread: all optimization strategies for large-scale clusters ultimately target system efficiency without sacrificing accuracy.

Scheduling strategies such as node and dynamic elimination mechanisms, while significantly boosting system efficiency through task differentiation, often come at the cost of reduced participation from long-tail data, given their compute-power-first filtering logic. Slow nodes that are frequently taken offline often harbor unique local features, making it easy to introduce systemic bias and undermine the generalization accuracy of the global model.

For correcting the client drift caused by data heterogeneity, proximal regularization and variance reduction provide rigorous convergence guarantees in mathematical theory. In practice, however, this theoretical accuracy preservation translates directly into extreme sensitivity to hyperparameters and a multiplied increase in communication overhead.

The error feedback mechanism, by introducing a memory buffer pool, perfectly compensates for the numerical noise introduced by gradient quantization, and serves as a critical line of defense for accuracy preservation under low bandwidth. Yet its limitation is equally stark: accumulating errors of the same dimensionality heavily squeezes limited GPU memory, which constrains its deployment in models with ultra-large-scale parameters.

A deeper analysis of these pain points and limitations points to one root physical cause: the irrationality of data partitioning. Within a distributed data parallel framework, the data partitioning strategy acts as the nexus connecting hardware topology with statistical algorithms. Early naive random data partitioning simply ignored the objective realities of cluster heterogeneity and non-IID data distributions. To truly achieve “accuracy preservation, “the key to breaking the impasse should not remain confined to post-hoc gradient compensation and task fault tolerance; it must be moved forward to the initial stage of data parallelism. Exploring a data partitioning paradigm that can dynamically assess data value and perform adaptive reorganization is the fundamental cure for the barrel effect and client drift.

Based on the above analysis, future breakthroughs in data partitioning and parallel strategies for accuracy-preserving distributed machine learning are likely to center on data-centric dynamic partitioning. This means moving beyond static data allocation patterns and building mechanisms for data migration and repartitioning grounded in real-time feedback. By evaluating the real-time training dynamics of each node, the system can replace passive waiting and discarding with dynamic data flow, achieving cluster load balancing and distributional homogeneity right from the data source.

5. Conclusion

This paper systematically delineates the research context of distributed machine learning in data partitioning methods, parallel coordination mechanisms, and accuracy-preserving strategies. The research shows that appropriate data partitioning, efficient parallel consistency and the fidelity of gradient transmission jointly determine the success or failure of distributed training. At the current stage, this field still faces notable gaps: accuracy-preserving strategies remain computationally expensive and communication compression algorithms lack sufficient theoretical support in extremely heterogeneous scenarios.

Future research should focus on three directions: adaptive partitioning algorithms for dynamic streaming data, novel data parallel topologies compatible with highly heterogeneous computing clusters, and software–hardware co-designed error compensation mechanisms that cover the entire chain from the data source to the aggregation endpoint. By these explorations, it is possible to build a large-scale distributed machine learning system that is both adaptive and accuracy-preserving.

References

[1] Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B.(2019). Megatron-LM: Training multi-billion parameter language models using model parallelism. arXiv preprint arXiv: 1909. 08053.

[2] Dean, J., & Ghemawat, S.(2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1), 107–113.

[3] Um, T., et al.(2024). Metis: Fast Automatic Distributed Training on Heterogeneous GPUs. 2024 USENIX Annual Technical Conference(USENIX ATC 24).

[4] Dean, J., & Barroso, L. A.(2013). The tail at scale. Communications of the ACM, 56(2), 74-80.

[5] Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., & Suresh, A. T.(2020). Scaffold: Stochastic controlled averaging for federated learning. International Conference on Machine Learning(ICML), 5132-5143.

[6] Um, T., et al.(2024). Metis: Fast Automatic Distributed Training on Heterogeneous GPUs. USENIX Annual Technical Conference(ATC).

[7] Alistarh, D., Grubic, D., Li, J., Tomioka, R., & Vojnovic, M.(2017). QSGD: Communication-efficient SGD via gradient quantization and encoding. Advances in Neural Information Processing Systems(NeurIPS).

[8] Qiao, A., Choe, S. K., Subramanya, S. J., Neiswanger, W., Peng, Q., Xing, E., &Ganger, G. R.(2021). Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning. In 15th USENIX Symposium on Operating Systems Design and Implementation(OSDI 21)(pp. 1-18).

[9] Peng, Y., Zhu, Y., Chen, Y., Bao, Y., Yi, B., Lan, C., ...&Guo, C.(2019). A generic communication scheduler for distributed DNN training acceleration. In Proceedings of the 27th ACM Symposium on Operating Systems Principles(SOSP '19)(pp. 16-29).

[10] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V.(2020). Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems(MLSys), 2, 429-450.

[11] Chai, Z., Ali, A., Zawad, S., Truex, S., Anwar, A., Baracaldo, N., ...&Yan, F.(2020). TiFL: A tier-based federated learning system. In Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing(HPDC)(pp. 125-136).

[12] Stich, S. U., Cordonnier, J. B., &Jaggi, M.(2018). Sparsified SGD with memory. Advances in Neural Information Processing Systems(NeurIPS), 31.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This paper is an output of the science project.

Open Access