Scalable Federated Learning for Scientific Foundation Models on Leadership-Class Systems

Olivera Kotevska
Oak Ridge National Laboratory
Oak Ridge, TN, USA
kotevskao@ornl.gov

Trong Nguyen
Oak Ridge National Laboratory
Oak Ridge, TN, USA

Rafael Ferreira da Silva
Oak Ridge National Laboratory
Oak Ridge, TN, USA
silvarf@ornl.gov

Christian Engelmann
Oak Ridge National Laboratory
Oak Ridge, TN, USA
engelmannc@ornl.gov

Prasanna Balaprakash
PrimaLabs
San Francisco, CA, USA
pbalapra@primalabs.ai

ABSTRACT

Federated learning (FL) at leadership-class HPC systems remains largely unexplored, despite growing interest in deploying federated workflows on modern HPC systems. This paper provides the first system-level empirical characterization of federated fine-tuning of pretrained foundation models on an exascale supercomputer under a multi-node deployment. Using up to 96 concurrent FL clients deployed across Frontier nodes, we study the impact of client scale, model size, data heterogeneity, partial participation, and differential privacy on runtime, communication overhead, and convergence stability.

Our results show that pretrained transformer models remain robust to heterogeneity, client dropout, and privacy noise, while system efficiency degrades rapidly with scale as synchronization and orchestration dominate runtime. We further demonstrate that system-aware execution strategies, including intra-node aggregation and early aggregation, significantly reduce wall-clock time without degrading model quality. These findings establish a practical performance baseline and inform the design of communication-efficient FL systems on leadership-class HPC platforms.

CCS CONCEPTS

• **Computing methodologies** → **Distributed computing methodologies**; **Machine learning**; **Distributed artificial intelligence**; • **Security and privacy** → *Security services*; *Privacy-preserving protocols*.

KEYWORDS

Federated Learning, Scalability, High-performance Computing, Foundation Models, Differential Privacy

1 INTRODUCTION

Federated learning (FL) enables collaborative model training across distributed data sources without centralizing raw data, making it attractive for scientific and institutional environments with strong governance and privacy constraints. While FL has been widely studied in simulation and small-scale cluster settings, real deployments on leadership-class HPC systems remain rare, particularly under realistic multi-node configurations. As a result, many assumptions about scalability and efficiency are derived from environments that do not reflect modern high-performance computing (HPC) systems.

At the same time, pretrained foundation models have become central to scientific machine learning workflows. Their strong representations enable rapid fine-tuning and improved robustness to data heterogeneity, making them natural candidates for federated training. However, their large parameter sizes and frequent synchronization requirements raise concerns about scalability and communication overhead when deployed across hundreds of nodes.

In this work, we study FL as a *distributed system*, rather than as an optimization algorithm. We present a system-level evaluation of federated fine-tuning of transformer-based models on a leadership class exascale supercomputer, using production-grade orchestration and realistic execution constraints. Our goal is not to introduce new FL algorithms, but to empirically characterize how existing approaches behave at scale and to identify the dominant system bottlenecks that emerge in practice. We emphasize that our experiments do not aim to saturate the full system capacity of Frontier (∼37K GPUs), but instead provide a controlled multi-node deployment (up to 96 GPUs) that captures key system-level behaviors (e.g., synchronization, orchestration, and communication bottlenecks) relevant to large-scale HPC environments.

Specifically, we address the following questions:

- How do communication and synchronization costs scale with client count and model size in federated training on exascale systems?
- How sensitive is system performance to data heterogeneity, client dropout, and differential privacy?
- Which system-level execution strategies most effectively mitigate bottlenecks in large-scale federated training?

By answering these questions through direct measurement on real hardware, this study complements prior simulation-based work and provides practical insights for the design of large-scale distributed ML systems.

2 BACKGROUND AND SYSTEM CONTEXT

FL enables collaborative training of a shared model across distributed clients without centralizing raw data. Each client performs local training and periodically communicates model

, ,
2026.

updates to a coordinating server, which aggregates and redistributes a global model. FL is widely used in settings where data governance, privacy, or institutional boundaries prevent centralized training.

Most existing FL studies focus on small- to medium-scale deployments, often relying on simulated clients or modest GPU clusters. As a result, the system-level behavior of FL at leadership-class scale remains poorly understood. Scaling FL to modern HPC systems introduces qualitatively different challenges, including thousands of concurrent clients, high-frequency synchronization, and tight coupling between computation, communication, and orchestration.

FL as a Distributed System. In this work, we study FL primarily as a distributed system rather than as an optimization algorithm. At scale, end-to-end performance is governed by communication latency, synchronization frequency, and orchestration overhead, often dominating local computation even on high-performance interconnects. These effects are difficult to capture in simulation or small-scale testbeds and require direct measurement on real hardware.

Foundation Models and Scalability. Pretrained foundation models are increasingly used in scientific machine learning due to their strong representations and rapid convergence during fine-tuning. In federated settings, they offer improved robustness to data heterogeneity and partial participation. However, from a systems perspective, large model sizes significantly increase communication payloads and synchronization cost. Understanding how model size interacts with client scale and system topology is therefore critical for federated deployment on HPC infrastructure.

Positioning and Scope. The goal of this study is not to introduce new FL algorithms, but to characterize how established FL methods behave under realistic, large-scale execution constraints. By evaluating federated fine-tuning on a leadership-class exascale system, we aim to identify dominant scalability bottlenecks and extract system-level insights relevant to the design of large-scale distributed ML systems.

Study	Year	Clients/GPUs used	Models/Dataset (size)
FedAvg [11]	2017	200 / NR	CNN, LSTM (NR); ∼60K–500K
LEAF [3]	2019	1000 / NR	LSTM (NR); ∼80K–1M
FLBench [16]	2020	66 / NR	NR; ∼80K–1M
FedML [6]	2020	112 / 8	ResNet-18 (11M), MobileNet (4M); ∼50K–200K
FedScale [8]	2022	10K / 10	ALBERT (12M), ResNet-34 (21M); ∼100K–10M
ORAF [10]	2022	5 / 6	ResNet-18 (11M), LSTM (NR); ∼10K–100K
APPFL v1.0 [7]	2022	NR / NR	CNN (NR); ∼50K–60K
pfl (Apple) [2]	2022	64 / 1	CNN (NR); ∼50K–200K
UniFed [4]	2023	894 / NR	CNN, RNN (NR); ∼60K–1M
MetisFL [13]	2024	200 / 0	CNN (NR); synthetic (∼10K)
FL Bench [14]	2024	10 / 10	CNN (NR); ∼5K–10K
FLHetBench [17]	2024	100 / NR	ViT-B (86M); ∼50K–1.2M
FL on HPC [15]	2024	6 / 6	CNN (NR); ∼60K–100K
This work	**2026**	**96 / 768**	**Transformers up to 185M; ∼12K (NCBI), ∼560K (DBpedia)**

Table 1: Comparison with prior FL studies in terms of deployment scale and workload characteristics. Studies that explicitly report both client counts and GPU resources typically correspond to real or partially real deployments, whereas studies reporting only client counts often rely on simulated clients. Client/GPU counts are reported as available in the original papers; NR = not reported.

3 RELATED WORK

FL has been studied extensively from an algorithmic perspective, with early work establishing convergence properties and communication-efficient optimization methods. These studies are typically evaluated using simulated clients or small-scale deployments and focus primarily on statistical performance rather than system behavior.

Several benchmarking frameworks have sought to increase realism by incorporating heterogeneous data distributions, partial participation, and device variability. However, as summarized in Table 1, existing evaluations remain limited in one or more key dimensions, including client scale, model size, hardware realism, or reporting detail. Most reported deployments involve at most hundreds of clients, relatively small models, or non-production orchestration, making it difficult to extrapolate their results to leadership-class systems.

More recent systems-oriented efforts have explored FL on GPUs and cloud infrastructure, providing valuable insights into distributed execution overheads. Nevertheless, these studies typically operate at modest scale or rely on simplified hardware and network configurations. In contrast, large-scale distributed training studies in the ML systems literature demonstrate that communication and synchronization dominate performance at scale, but largely focus on centralized training and do not capture the stricter synchronization semantics and partial participation inherent to FL.

This work complements prior FL and ML systems research by providing a system-level empirical characterization of FL deployed on a leadership-class exascale supercomputer. By evaluating federated fine-tuning of foundation models under production orchestration, we directly measure communication, synchronization, and orchestration costs at scale and establish a reference point for future large-scale federated deployments.

To the best of our knowledge, this is the first study to evaluate federated fine-tuning of transformer-based foundation models on a leadership-class exascale system under production orchestration. Unlike prior work, our contribution lies not in saturating the full hardware scale, but in providing a

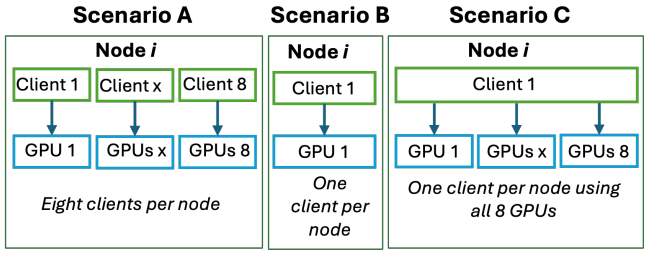


Figure 1: Client-to-GPU mapping strategies. Left: Scenario A assigns one client per logical GPU within a node (eight clients per node). Middle: Scenario B assigns one client to a single logical GPU per node. Right: Scenario C assigns one client to all logical GPUs within a node via intra-node data parallelism.

controlled, production-grade multi-node deployment on a leadership-class system that exposes system-level bottlenecks not observable in small clusters or simulations.

4 EXPERIMENTAL APPROACH AND DESIGN

This section describes the methodology used to evaluate FL at scale on a leadership-class HPC system. We first outline the hardware and software environment, followed by the experimental design used to study scalability across clients, models, and execution strategies.

4.1 Experimental Setup

Hardware. Experiments were conducted on Frontier, an exascale supercomputer at Oak Ridge National Laboratory [1]. Each node comprises AMD EPYC CPUs and four AMD Instinct MI250X GPUs connected via high-bandwidth intra-node links, with inter-node communication over the HPE Slingshot-11 network. Each MI250X comprises two GPU compute dies (GCDs). Under ROCm, Frontier exposes eight logical GPU devices per node. Throughout this paper, 'GPU' refers to these logical devices.

Software. Federated orchestration uses NVIDIA NVFlare in production mode [12], with local training implemented in PyTorch (ROCm) on AMD Instinct MI250X GPUs. Experiments are launched via a custom scheduler-integrated automation framework to enable scalable client placement and reproducible execution. Unless otherwise noted, all runs use 10 communication rounds, 5 local epochs, batch size 16, and a fixed learning rate, a configuration chosen to isolate system-level behavior in federated fine-tuning.

4.2 Experimental Design

Our experimental design evaluates FL along three primary dimensions; across all experiments, we measure both learning outcomes and system-level metrics.

4.2.1 Client-to-Resource Mapping. To assess how execution topology affects scalability, we evaluate three client-to-GPU mapping strategies illustrated in Figure 1:

- **Scenario A**: One client per GPU within a node, resulting in eight clients per node.
- **Scenario B**: One client per node, with each client assigned to a single GPU.
- **Scenario C**: One client per node utilizing all GPUs within the node via intra-node data parallelism.

Scenario A maximizes client density within a node, stressing local concurrency. Scenario B isolates clients across nodes to emphasize inter-node communication and synchronization costs. Scenario C represents an HPC-style deployment in which each client leverages intra-node parallelism, reducing the number of cross-node synchronization events. Together, these scenarios enable analysis of how client placement and locality influence FL performance at scale.

4.2.2 Foundation Models and Datasets. We evaluate pretrained transformer models spanning a range of sizes, from lightweight architectures to models with up to 185M parameters, to study the impact of model scale on federated training. Experiments use two text classification datasets: a small biomedical dataset (NCBI Disease Corpus [5]) and a larger general knowledge dataset (DBpedia ontology types [9]). Unless otherwise noted, BERT-base trained on the biomedical dataset is used as the default configuration for scalability experiments.

4.2.3 Data Heterogeneity. To assess robustness under non-IID data, datasets are partitioned across clients using Dirichlet distributions with varying concentration parameters. Results are reported for IID and three non-IID settings. Here, data heterogeneity primarily serves as a system stressor, enabling analysis of its effect on runtime and synchronization rather than algorithmic convergence.

4.2.4 Client Dropout. To emulate execution variability at scale, we introduce controlled client dropout, where a fixed fraction of clients is randomly excluded in each round. We evaluate two dropout-tolerant aggregation policies that proceed once a predefined fraction of updates is received, allowing us to study the impact of partial participation and early aggregation on runtime and convergence stability.

4.2.5 Differential Privacy. We evaluate federated training under differential privacy using Gaussian noise applied to local updates, with two privacy budgets. The analysis focuses on system-level overhead, particularly the interaction between DP-induced computation and communication-dominated execution at scale, rather than on formal privacy guarantees.

4.3 Performance Metrics

Federated training is evaluated using both learning and system-level metrics. Learning quality is measured by final model accuracy, while system efficiency is assessed through total runtime, communication time, and per-round execution cost. Together, these metrics characterize FL as both a learning process and a distributed system deployed at leadership scale.

5 RESULTS AND DISCUSSION

Across all evaluated configurations, model accuracy stabilizes within the early communication rounds, consistent with the rapid convergence of pretrained foundation models under federated fine-tuning. We observe no accuracy degradation or instability across scaling scenarios once early-round convergence is reached, indicating that the reported final accuracies reflect stable training behavior rather than transient effects.

5.1 Baseline: Independent and Identically Distributed Data (IID)

We first establish a baseline using IID data across eight clients. Under IID (random) partitions, all three aggregation methods (FedAvg, FedProx, FedOpt) converge stably with accuracies exceeding 99% across all client-to-GPU mappings (Table 2). In contrast, runtime and communication costs vary substantially by execution topology (Figure 1).

Scenario C achieves the lowest end-to-end runtime, benefiting from intra-node data parallelism within each client. Scenario B ranks second but incurs higher latency due to inter-node synchronization over Slingshot–11. Scenario A is consistently slowest, reflecting higher per-round overhead when many clients share a node. Overall, under IID data, algorithm choice has minimal impact on accuracy; *system topology is the primary determinant of efficiency.*

5.2 Impact of Data Heterogeneity

To evaluate robustness under statistical skew, we train with Dirichlet partitions with $\alpha \in \{0.9, 0.6, 0.3\}$, where smaller α indicates stronger non-IID behavior. Table 2 reports accuracy, communication time, and total runtime for all scenarios relative to the IID baseline.

Accuracy remains high across all heterogeneity levels, with only modest degradation under the strongest skew. The primary effect of heterogeneity is on system efficiency: Scenario B experiences the largest runtime inflation as even mild non-IID behavior amplifies waiting and synchronization effects when clients are distributed across nodes. Importantly, the reported increase in 'communication time' reflects not only network transfer but also synchronization overhead and barrier waiting time. Because model sizes and update payloads remain constant across IID and non-IID settings, the underlying data transfer cost is unchanged. The observed increase is therefore dominated by straggler effects, where variability in local training time under heterogeneous data distributions leads to increased idle waiting at synchronization points. In contrast, Scenarios A and C benefit from intra-node locality, which mitigates synchronization sensitivity despite similar statistical conditions. We note that the conservative training configuration used here intentionally emphasizes clean system-level measurement and may suppress late-phase failure modes (e.g., optimizer drift under long horizon training). Investigating such effects remains important for training from scratch or longer horizon regimes.

In summary, increasing data heterogeneity has limited impact on accuracy but can substantially increase runtime when communication is inter-node dominated (Scenario B), reinforcing that synchronization not statistical skew is the dominant bottleneck at scale.

For clarity, the reported communication metric includes both message transfer and synchronization-induced waiting time at round barriers; under non-IID splits, increases in this metric therefore mainly reflect straggler effects rather than larger payloads.

5.3 Scalability and Strong-Scaling Analysis

We next study how federated training scales with the number of clients. Unless otherwise noted, we focus on Scenario B with FedAvg and IID splits; Scenarios A and C show qualitatively similar trends.

Accuracy vs. number of clients. Accuracy remains high from 8 to 96 clients (96.0–100.0%), with the best result observed at full scale (Table 3). This indicates that scaling to nearly one hundred nodes does not degrade model quality in this fine-tuning regime.

Strong scaling. Table 3 and Figure 2 show that speedup remains below one for $n > 8$ and parallel efficiency drops sharply with scale. Note that in Figure 2 the y-axis is labeled "Value" and jointly represents speedup and efficiency (fractional), due to plotting constraints; we therefore report exact numerical values in Table 3 for clarity. As shown in Table 3, efficiency drops rapidly (to 2% at 96 clients), highlighting the strong impact of synchronization overhead at scale. Because the global dataset size is fixed, adding clients increases synchronization and orchestration overhead faster than computation decreases, causing Scenario B to become communication-bound.

Runtime decomposition. Communication dominates at scale: moving from 8 to 96 clients increases per-round communication from $\sim$33 s to $\sim$275 s, while local training time decreases modestly. Thus, the growth in T_n is driven primarily by synchronization over Slingshot–11 rather than GPU compute, motivating topology- and communication-aware strategies (e.g., Scenario C or hierarchical aggregation).

In all experiments, the number of local epochs is kept constant across different client counts. As the total dataset is partitioned across more clients, the number of local iterations per client decreases proportionally. This reduces per-client computation while keeping communication frequency fixed, thereby shifting the compute-to-communication ratio toward communication-dominated execution at larger scales.

5.4 Impact of Model Size

We next evaluate the effect of model size at full scale (96 clients) using three pretrained transformers: ALBERT (12M), BERT (110M), and DeBERTa (185M). Results are aggregated across FedAvg, FedProx, FedOpt and data distributions. Table 4 shows that communication cost scales strongly with model size (ALBERT $\sim$130 s/round; BERT $\sim$276 s/round; DeBERTa $\sim$430 s/round), while training time per round increases slowly. Figure 3 further shows that communication grows roughly linearly with client count with a model-size-dependent offset. Accuracy remains high across scales, with larger models

Table 2: Performance comparison for IID and Dirichlet distributed data at 8 clients. "Comm." denotes aggregate communication-related overhead, including synchronization-induced waiting time at round barriers. IID yields the highest accuracy; Scenario A and C are fastest; Scenario B incurs higher latency due to inter-node communication and synchronization. Runtime increase under heterogeneity is most pronounced in Scenario B.

Scen.	Alg.	Random			$\alpha = 0.9$			$\alpha = 0.6$			$\alpha = 0.3$		
		Acc.	Comm.	Time	Acc.	Comm.	Time	Acc.	Comm.	Time	Acc.	Comm.	Time
A	FedAvg	99.64	39.23	868.12	99.03	25.39	375.20	99.24	28.03	400.50	99.07	26.63	389.58
A	FedProx	99.54	39.52	913.73	98.95	26.59	396.70	99.32	29.49	425.70	98.58	27.37	399.00
A	FedOpt	99.69	40.17	879.22	99.31	25.41	377.73	99.34	28.07	399.15	99.43	26.13	380.43
B	FedAvg	99.62	32.94	749.66	99.36	62.88	1057.35	99.63	75.91	1183.76	99.12	63.85	1067.05
B	FedProx	99.54	31.88	777.69	99.34	73.73	1245.50	99.63	88.02	1389.00	99.07	75.97	1268.62
B	FedOpt	99.79	32.35	743.11	99.62	63.71	1070.70	99.71	72.23	1164.51	99.38	63.46	1063.20
C	FedAvg	99.51	31.69	716.28	98.98	24.95	364.94	99.02	29.04	396.08	98.73	25.56	361.36
C	FedProx	99.69	31.67	763.59	99.13	25.74	376.79	99.22	29.40	412.01	98.68	27.37	393.45
C	FedOpt	99.72	33.13	732.23	99.23	25.09	361.00	99.44	27.93	391.91	99.07	25.65	366.36

Table 3: Strong-scaling of Scenario B (FedAvg, random split) on *Frontier*, using the 8-client run as baseline. $S_n = T_8 T_n$, $E_n = S_n n8 \times 100\%$.

#Clients	T_n (s)	S_n	E_n (%)	Acc. (%)
8	749.66	1.00	100.0	99.62
24	948.41	0.79	26.3	98.98
48	1559.44	0.48	8.0	97.56
72	2195.94	0.34	3.8	96.04
96	3103.81	0.24	2.0	100.0

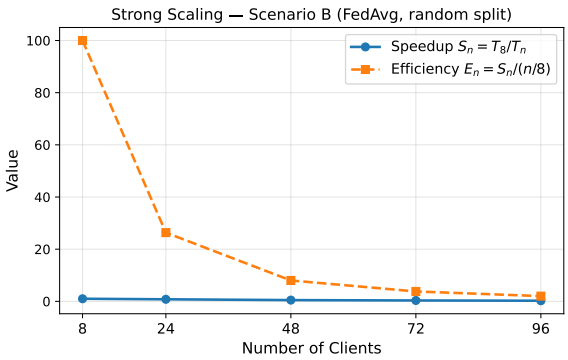


Figure 2: Strong-scaling of Scenario B (FedAvg, random split) on Frontier relative to the 8-client baseline. The solid line shows speedup $S_n = T_8 T_n$. The dashed line shows parallel efficiency $E_n = S_n n8$. Note: the y-axis is labeled "Value" due to plotting constraints; it represents both speedup and efficiency (as a fraction in the range 0–1). Efficiency can be interpreted as a percentage via $E_n\% = 100 \times S_n n8$.

slightly more robust under stronger non-IID partitions. We

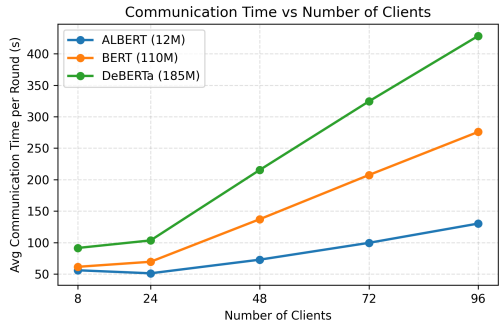
Table 4: Average communication time and accuracy at 96 clients for ALBERT, BERT, and DeBERTa. *Values are averaged over FedAvg, FedProx, and FedOpt and all data distributions.*

Model	Params (M)	Comm. (s)	Acc. (%)
ALBERT	12	∼130	90–99
BERT	110	∼276	97–100
DeBERTa	185	∼430	97–100

also observe a non-monotonic behavior at intermediate client counts, most visibly around 48 clients in Figure 3(b), where accuracy drops before recovering at larger client counts. A plausible explanation is that intermediate scaling regimes expose competing effects: each client receives a smaller local partition, which increases gradient variance and sensitivity to partition imbalance, while synchronization overhead also grows. At larger client counts, the larger aggregation pool appears to stabilize the global update, leading to partial recovery in accuracy. This phenomenon is not commonly discussed in small-scale or simulated FL studies and highlights the importance of measuring real multi-node deployments. Overall, these results confirm that for federated fine-tuning on exascale systems, *network communication is the primary limiter* for larger foundation models, motivating compression, sparsification, partial updates, or hierarchical aggregation.

5.5 Impact of Dataset Size

To assess how workload size shifts system behavior, we compare NCBI (∼12k samples) to DBpedia (∼560k samples) at 96 clients. Table 5 shows that the larger dataset substantially increases total runtime (compute-bound shift), while communication increases only moderately because update size is independent of dataset size.



(a) Communication time per round vs. number of clients for ALBERT, BERT, and DeBERTa.

(b) Test accuracy vs. number of clients for the three model sizes.

Figure 3: Effect of model size on FL scalability. Each curve averages over FedAvg, FedProx, and FedOpt and all data distributions. Communication cost scales roughly linearly with client count and proportionally with model size, while accuracy remains high for all three transformers.

Table 5: Performance comparison on small (NCBI) vs. large (DBpedia) dataset at 96 clients. *DBpedia is approximately* 46× *larger than NCBI. Runtime grows substantially due to increased computation, while communication grows only moderately.*

Alg.	NCBI			DBpedia		
	Acc.	Comm.	Total	Acc.	Comm.	Total
FedAvg	100.0	275.3	3103.8	99.17	481.4	5367.1
FedProx	99.69	272.8	3069.6	99.04	493.4	5605.8
FedOpt	99.07	275.1	3103.2	99.18	475.9	5415.2

These results highlight a regime change: with small datasets, federated fine-tuning is communication-dominated; with larger datasets, local compute becomes a larger fraction of the end-to-end cost, improving effective utilization while preserving high accuracy.

Table 6: Comparison of Dropout-Tolerant Scenarios B-1 (90% threshold) and B-2 (80% threshold) at 96 clients. Values are averaged across Dirichlet $\alpha \in \{0.9, 0.6, 0.3\}$ and IID splits. Scenario B-2 reduces total runtime by 15% relative to B-1 while maintaining nearly identical accuracy ($<$ 0.3% difference).

Scen.	Alg.	Acc. (%)	Time (min)	Comm. (s)	Train (s)
B-1	FedAvg	98.2	42	270	2.3
	FedProx	98.3	41	268	2.2
	FedOpt	**98.5**	**39**	**260**	**2.1**
B-2	FedAvg	98.1	38	255	2.0
	FedProx	98.2	37	253	1.9
	FedOpt	**98.4**	**35**	**248**	**1.8**

5.6 Asynchronous Dropout-Tolerant Training

We extend Scenario B to two dropout-tolerant variants at 96 clients. In both cases, clients that miss the aggregation threshold in a round are skipped for that round and rejoin later (no cross-round update merging). Scenario B-1 aggregates when 90% of client updates have been received. Scenario B-2 triggers earlier aggregation at 80% and employs more aggressive overlap between communication and computation.

Table 6 shows that B-2 reduces runtime by ∼15% relative to B-1 while maintaining nearly identical accuracy. These gains come from reduced idle waiting and improved overlap, indicating that partial participation policies are a practical systems lever for mitigating stragglers at scale.

Note: Train (s) in Table 6 reports non-overlapped local training time per round measured at the client, excluding preprocessing and any DP operations.

5.7 Impact of Differential Privacy

Note that training time metrics in this section are not directly comparable to those in Section 5.6 due to DP-specific computation overheads. We evaluate DP overhead at full scale (96 clients) using BERT under three settings: no DP, DP with $\varepsilon = 1$, and DP with $\varepsilon = 3$ using NCBI dataset. Table 7 summarizes average training time, communication time, total runtime, and accuracy.

Communication time remains nearly constant across privacy settings, as DP does not change message size. DP increases local training time due to clipping and noise injection, but because execution at scale is communication-dominated, the impact on end-to-end runtime remains modest. Accuracy remains high across privacy budgets, indicating that privacy-preserving federated fine-tuning is feasible without substantially exacerbating scalability bottlenecks.

5.8 Implications for FL Evaluation

Beyond answering the experimental questions posed in Section 1, our results also clarify which assumptions from prior FL evaluations hold in realistic multi-node HPC deployments and which do not.

Table 7: Comparison of differential privacy (DP) overheads at 96 clients. Values are averaged over data distributions.

Setting	Train (s)	Comm. (s)	Total (s)	Acc. (%)
No DP ($\varepsilon = \infty$)	8.8	275	2920–3100	95.6–100.0
DP ($\varepsilon = 1$)	10.3	276	2890–2965	95.9–100.0
DP ($\varepsilon = 3$)	10.5	275	2898–2972	95.9–100.0

First, our measurements show that system overhead at scale is not explained by payload size alone. In several settings, the dominant overhead is synchronization-induced waiting rather than raw data transfer, especially under statistical heterogeneity where local training times diverge across clients. This distinction is often obscured in simulation-based studies that report communication as a single aggregate quantity.

Second, we find that non-IID data has limited effect on final accuracy in our fine-tuning regime, but can substantially increase end-to-end runtime through straggler amplification. This suggests that, in production FL settings, data heterogeneity should be treated not only as a statistical challenge but also as a systems challenge.

Third, our results indicate that small testbeds and simulated clients may be sufficient for studying convergence trends, but can underestimate synchronization, orchestration, and placement effects that dominate runtime on real systems. In particular, the relative benefits of intra-node aggregation and early aggregation policies only become clear when these system-level effects are measured directly.

Overall, these findings suggest that realistic multi-node deployments are necessary to accurately characterize the scalability limits of federated learning systems, even when the learning dynamics themselves appear stable.

5.9 Summary of System-Level Takeaways

Across all experiments, three trends are consistent. (i) Federated fine-tuning remains stable and accurate at scale, even under non-IID data, partial participation, and DP noise. (ii) System topology (Figure 1) strongly determines efficiency: inter-node synchronization dominates runtime as client count and model size increase. (iii) Communication-aware execution strategies, including intra-node aggregation and early aggregation thresholds, offer substantial efficiency gains without harming convergence.

6 CONCLUSION

This paper presents a system-level empirical study of FL deployed on a leadership-class exascale supercomputer. Through large-scale experiments with pretrained foundation models, we show that federated training remains accurate and stable at scale, but rapidly becomes communication-dominated due to synchronization and orchestration overheads. While our experiments utilize a small fraction of Frontier's total capacity, they reveal system-level behaviors that are expected to amplify at larger scales, particularly synchronization and orchestration overheads.

We demonstrate that system-aware execution strategies, including intra-node aggregation and early aggregation under partial participation, significantly improve efficiency without degrading model quality. These results suggest that future scalability gains in FL on HPC systems will depend more on communication-aware system design than on algorithmic modifications alone.

Together, these results answer the central questions posed in the introduction by demonstrating that federated fine-tuning of foundation models is statistically robust at leadership scale, but fundamentally constrained by communication and orchestration overheads—constraints that can be mitigated through system-aware execution strategies.

REFERENCES

[1] [n. d.]. Frontier: The First Exascale Supercomputer — OLCF / Oak Ridge National Laboratory. https://www.olcf.ornl.gov/frontier/. Accessed: 2025-12-04.

[2] Apple. 2022. pfl: Python Framework for Private Federated Learning Simulations. https://github.com/apple/pfl-research.

[3] Sebastian Caldas, Satyajit Duddu, Peter Wu, Tian Li, Jakub Konecny, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2019. LEAF: A Benchmark for Federated Settings. *arXiv preprint arXiv:1812.01097* (2019).

[4] Xiaojie Chen, Chengliang Liu, Weizhong Lin, Qifan Chen, Pan Zhou, and Xiaohui Li. 2023. UniFed: A Unified Benchmark for Federated Learning. *arXiv preprint arXiv:2303.08952* (2023).

[5] Rezarta I. Dogan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: a resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics* 47 (2014), 1–10. https://doi.org/10.1016/j.jbi.2013.12.006

[6] Chaoyang He, Shiqiang Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, William Fedus, Ramesh Raskar, and Qiang Yang. 2020. FedML: A Research Library and Benchmark for Federated Learning. *arXiv preprint arXiv:2007.13518* (2020).

[7] Minseok Kim, Aman Sharma, Fei Yin, and Kibaek Park. 2022. APPFL v1.0: Privacy-Preserving Federated Learning Framework. *arXiv preprint arXiv:2202.03672* (2022).

[8] Fan Lai, Yihan Dai, Xiangfeng Wang, Xun Zhang, Chaoyang He, Xu Wang, Jian Liu, Bo Li, Ce Zhang, Yongxin He, Lichao Sun, and Mi Zhang. 2022. FedScale: Benchmarking Model and System Performance of Federated Learning at Scale. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*. 11814–11827.

[9] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. In *Semantic Web Journal*, Vol. 6. 167–195. https://doi.org/10.3233/SW-140134

[10] Tian Li, Chulin Xu, Andrew Hard, Soheil Kim, and Virginia Smith. 2022. ORAF: An Open Federated Learning Benchmark Suite. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*. ACM.

[11] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 1273–1282.

[12] Quoc Nguyen, Karthik Sreenivasan, Yizhe Chen, Julius Wicaksana, Shuang Xu, Luyi Yang, Qing Liu, Yu Wang, Holger Roth, Dan Xu, Ziyue Xu, Ziyue Liu, Daguang Xu, and Li Shen. 2022. NVIDIA FLARE: Federated Learning from Simulation to Real-World. *arXiv preprint arXiv:2202.03341* (2022). https://arxiv.org/abs/2202.03341

[13] Nikolaos Papadopoulos, Stefanos Laskaridis, Stylianos Venieris, Christos Strydis, Nicholas D. Lane, and George Karypis. 2024.

MetisFL: An Elastic and Fair Federated Learning Framework. In *Proceedings of the 19th European Conference on Computer Systems (EuroSys)*. ACM.

[14] B Pfitzner, JM Köhler, L Schilling, S Gaube, K Schramm, J Franke, and KH Maier-Hein. 2024. Benchmarking Federated Learning Frameworks for Medical Imaging Tasks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 222–234.

[15] Jialin Song, Zhihao Li, Ananya Gupta, Wei Xu, and Yuxiong Liu. 2024. Benchmarking Federated Learning on High-Performance Computing Systems. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE.

[16] Weiming Zeng, Tian Zhou, Bin Luo, and Yong Liu. 2020. FLBench: A Benchmark Suite for Federated Learning. *arXiv preprint arXiv:2008.07257* (2020).

[17] Zonghan Zhang, Xiang Chen, Di Wu, Jianyu Wang, Ming Li, and Yuxuan Wang. 2024. FLHetBench: Benchmarking Device and State Heterogeneity in Federated Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.