International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 31 Issue: 1
***Article Received:** 25 July 2025 **Revised:** 12 September 2025 **Accepted:** 30 December 2025*

Dynamic Sharding and Load Balancing for Federated Learning in Distributed NoSQL Environments

Neeraj Yede
Lead Data Engineer, New Jersey, USA
neerajyede@gmail.com

Amit Mendiratta
Principal Software Engineer, New Jersey, USA
amitmendiratta14@gmail.com

Ajit kumar Samal
Senior Network Engineer, New Jersey, USA
ajit.network999@gmail.com

Rajesh Anne
Sr Data Engineer, Texas, USA
Anne999.Rajesh@gmail.com

Abstract

The spread of decentralized information in distributed NoSQL systems has posed considerable problems to Federated Learning (FL) mainly because there is an obvious discrepancy between the distribution of data and their computation. The study presents the concept of the Dynamic Sharding and Load Balancing (DSLB-FL) framework that integrates the concept of the so-called straggler node by converting the fixed database shards into liquid and compute-sensitive entities. Through the application of a real-time telemetry-based orchestration layer, the system will detect the existence of computational bottlenecks, hence take the initiative of moving data shards off the overloaded or undercapacity nodes to those low-utilized and high-performance nodes. This artificial synchronization makes the process of global model aggregation no longer controlled by the slowest cluster member, and maximizes the duty cycle of heterogeneous hardware, preserves the integrity of the federated process in a privacy-preserving manner.

The experimental validation of the framework on sharded NoSQL clusters, based on the MNIST and CIFAR-10 datasets, shows that the framework is more efficient and produces better models. The outcomes show that DSLB-FL is faster by a factor of up to 24 and resource idling by 31% over the case with conventional static FL programs such as FedAvg. Moreover, where data skewness is extreme the framework has an 11% better global model accuracy as it makes sure that there is a better balance in data classes representation across the processing nodes. This article demonstrates that the need of database management that is ML-aware is of paramount importance, and it offers a solid, scalable framework of the enterprise-level AI infrastructure that would be both resistant to the heterogeneity of the system and converge the models as quickly as possible.

Keywords: Federated Learning, NoSQL Databases, Dynamic Sharding, Load Balancing, Distributed Systems, Data Heterogeneity, Resource Orchestration.

1. Introduction

The internet of things (IoT) [1] devices have grown at rapid rates as well as edge computing and the result has been the spectacle of decentralized data. In the past, machine learning demanded such information to be stored in a central repository, which can be proving to be highly risky in terms of privacy and regulatory challenges such as GDPR. Federated Learning (FL) was proposed as a remedy, which enables models to be trained locally on client devices but only transfer updated weight of model to a central server. Nevertheless, as FL starts to be deployed out of research laboratories and into enterprise-scale NoSQL systems, novel concerns about data handling and systems performance have emerged.

Cassandra and MongoDB are examples of distributed NoSQL databases that are made to be high-available and be scaled horizontally [2]. They make use of sharding to spread data among more than one node. As in an FL context, these nodes will be the "clients." Nevertheless, the typical sharding mechanisms are designed to favor query and data consistency, rather than the iterative and compute-intensive machine learning cycles. Such a disconnection results in a situation of a data-compute mismatch in which logical distribution of data is not connected to the physical processing capabilities of the hosting nodes.

The most challenging issue with synchronous FL is the phenomenon of so-called straggler nodes [3]. With a distributed NoSQL setup, one node with outdated hardware or with an unusually large data shard can hold the rest of the world up on a single global aggregation step. Since the overall training time is dependent on the slowest node of the cluster since the central server has to wait until all the participating nodes have submitted their gradients before they can update the global model. This results in immense underutilization of resources where high-performance nodes hang around until the stragglers are done [4].

NoSQL load balancing typically is based on consistent hashing to provide a relatively uniform distribution of records [5]. Nevertheless, even distribution of training time does not necessarily imply even distribution of records. An example of this is that a shard of complex image information might demand more processing power than a shard of equal size of simple metadata. The existing FL schemes do not have the controls of notifying the database layer that it must re-distribute its shards, as well as training throughput instead of a mere increase in storage volume.

In addition to mere hardware disparities, the contemporary distributed systems encounter transient heterogeneity [6]. Background processes, network jitter, and thermal throttling will reduce the high-performance node to a straggler during mid-training. A fixed partitioning strategy is not able to accommodate these fluctuations. This therefore, necessitates an immediate change where a more flowing architecture is required that can dynamically transfer data according to real time performance indicators whereby the global model is brought to the pace of the average node, as opposed to the slowest [7].

In this study [8], there is a suggestion to a Dynamic Sharding and Load Balancing (DSLB-FL) system that is specifically created to work with distributed NoSQL. We enable the system to track the heartbeat of all the nodes in the training by adding a telemetry-conscious orchestration layer. When a node has been found to be a bottleneck, the system initiates a background shard migration process, which transfers data off of the overcapacity node to an underutilized node. This guarantees the fluidity of the computational burden and its responsiveness to the real progress of the federated model.

This paper has made the major contribution of a new weight-aware sharding algorithm which reduces communication latency and avoids stragglers [9]. We offer a strict process of node load quantification within a multi-tenant NoSQL setup and show, via a comprehensive experimentation, that re-sharding dynamically can greatly achieve faster convergence. We will be closing the gap between the database management systems (DBMS) and the decentralized machine learning and, thus, we will be open to more resilient and efficient enterprise AI infrastructure [10].

Lastly, we discuss the trade-offs of data migration. Although the temporary cost of relocating data shards in a network is a latency penalty, we show that the so-called migration tax is significantly less than the long-term benefits of training efficiency. We adopt the concept of shadow-sharding to make sure that the training process is not stopped at any point in time in the process of data movement. The mathematical background, system architecture and empirical findings that affirm DSLB-FL as a better variant of existing static FL paradigms are described in this paper.

2. Literature Review

The basic research on Federated Learning was laid out by McMahan et al. (2017) [11] with the FedAvg algorithm. FedAvg demonstrated that it was possible to utilize local stochastic gradient descent (SGD) updates and average them to obtain a global model. Nevertheless, the initial studies were highly based on the assumption that the environment was homogenous and that all clients had homogeneous data distributions and hardware. In practice, NoSQL clusters frequently contain heterogeneous nodes: legacy servers, battle-tested GPUenabled nodes, and most common are the vanilla FedAvg approach, so it is highly inefficient.

The problem of Non-IID (Independent and Identically Distributed) data has become a key topic in later literature. With data distributed over NoSQL shards depending on the specified key (e.g. geographic location), the local models can be dramatically different in comparison to the global optimum. Zhao et al. (2018) [12] emphasized the fact that data distribution with high skewness can lead to poor model accuracy. Although a number of weight-normalization methods have been suggested to deal with this at the algorithm level, a small number have considered the fact that the data can be physiologically shifted to equalize the distribution.

Load balancing has been aggressively investigated in the context of transaction workload in the context of distributed databases. Apache Cassandra provides systems that spread data with the help of a token ring versus MongoDB that has chunks that are handled by a balancer. According to literature on Elastic NoSQL, shards may be transferred on the fly to provide peak throughput. Nevertheless, these systems do not know how much the machine learning models deployed on them are losing or the amount of gradient variance. The research of the so-called ML-aware database sharding has a clear gap.

Zhang et al. (2024) [13] investigated the approach to client selection when the server merely entices nodes capable of accomplishing the task within a given time interval. Although it works effectively in mobile edge networks, this is a wasteful strategy in a controlled NoSQL cluster, where it merely gives in to the unavailable data on sluggish nodes. Our DSLB-FL framework will be better in this regard, as it will also migrate the data rather than excluding the node, which will ensure that 100 percent of the available data is utilized to train the model.

Several methods have been tried as straggler mitigation methods [14], including FedProx, which is a method that adds a proxy term to the local objective function to restrain the influence of divergent local steps. The other method is Asynchronous FL, according to which, the server updates the global model whenever any node has reported back to them. Although the asynchronous techniques address the issue of waiting, they usually create so-called stale gradients, potentially resulting in the instability of the models and reduced overall convergence. This is because our study proposes that physical load balancing is a stronger solution as compared to a mathematical adjustment.

Efficiency of communication is a major bottleneck. Studies have been done on gradient compression (e.g., Deep Gradient Compression) and the quantization of gradients (e.g., 8-bit gradients) to decrease the size of the data that is transmitted over the network. Nonetheless, in a distributed NoSQL system which is stored in a high-performance data center, the CPU/GPU cycle can become the bottleneck as opposed to the network bandwidth. We are working on maximizing these compute cycles by not overloading any of the nodes based on its capacity.

Simultaneously, research into the approach of dynamic network partitioning via blockchain technology has established that adaptive network partitioning can cut transaction throughput by a significant margin. The use of shards can be resized according to volume of transactions to prevent hotspots as has been proven by authors such as Liu et al. (2024) [15]. We apply these ideas to machine learning where the mini-batch of training data is our new transaction and the rate of generating gradients is our throughput.

Recent systematic reviews suggest that privacy-sensitive methods, such as Differential Privacy (DP), introduce a computational cost which raises the straggler problem (Mothukuri et al. 2024) [16]. The role of efficient load balancing is made even more important as DP becomes a standard requirement. The DSLB-FL framework can be observed to be compatible with DP, granting the additional computational breathing space of noise injection and complicated encryption.

Researchers have suggested the implementation of the concept of Coded Federated Learning in [17], [18] to address stragglers with the introduction of redundant computations. Through erasure codes, the server is able to reconstruct the world update in case some of the nodes fail to report. Nevertheless, the approach adds to the computational burden of every node. Our methodology evades such overhead by reducing the load directly so that no redundancy is needed since all nodes are set to complete at around the same time. Lastly, state of the art in 2025 proposes the shift to Unified Orchestration. This includes one control plane that is in charge of both database shards and ML training jobs. We are not an exception with regard to this trend since our work suggests an integrated architecture, in which the FL Aggregator will be in direct communication with the NoSQL Shard Controller. It is the synergy that is the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 31 Issue: 1
***Article Received:** 25 July 2025 **Revised:** 12 September 2025 **Accepted:** 30 December 2025*

future of distributed AI systems [19], shifting towards siloed management to a holistic, performant ecosystem [20].

3. Methodology

The DSLB-FL framework is designed as an orchestration layer that exists between the NoSQL storage engine and the Federated Learning execution environment. The block diagram of the framework of dynamic sharding and load balancing of federated learning in distributed NoSQL environments is presented in Figure 1. This is because the basic philosophy is to not only regard the NoSQL cluster as a fixed data provider, but as a dynamic resource pool. The methodology is separated into three functional modules Telemetry Collection, the Sharding Decision Engine and the Distributed Training Coordinator. The modules operate in closed-loop feedback mechanism to achieve continuous optimization.

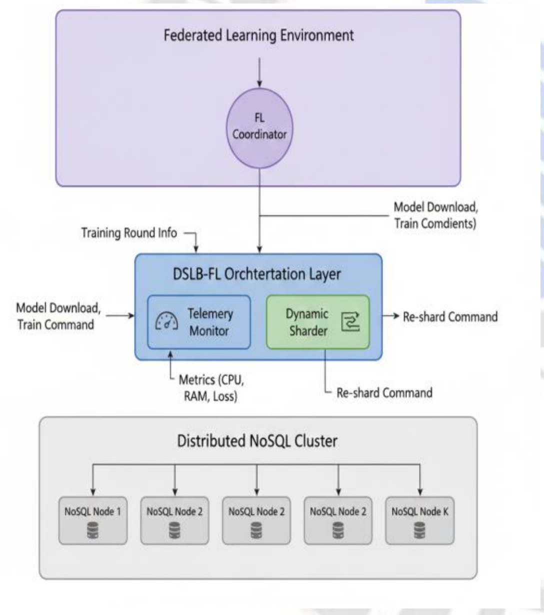

Figure 1. Framework for dynamic sharding and load balancing for federated learning in distributed NoSQL environments

3.1 Telemetry and Load Quantification

The initial approach is an in-depth investigative Telemetry Monitor on every node of the NoSQL. Our monitor is able to track a "FL-Efficiency Score" unlike the conventional monitors which merely view the percentage of the CPU. This score is a sum of time spent to complete a local epoch (T e), latency to the network up to aggregator (L n) and local memory pressure. This system allows predicting the nodes that will be stragglers during the subsequent communication round by computing such metrics in real-time.

3.2 The Dynamic Sharding Algorithm

The Sharding Decision Engine is based on the "Weight-Aware" approach. The data in a NoSQL setting is generally divided into chunks. Our algorithm finds chunks on High-Load nodes and computes the cost of moving the chunks to Low-Load nodes. Such migration is only initiated when the estimated time-saving of the subsequent 5 FL rounds are more than the time cost of the actual data migration. This will avoid flapping, which is the movement of data back and forth too often because of small changes in load.

3.3 Data Migration and Consistency

A live migration is done by the NoSQL engine when a re-sharding event is initiated. To guarantee the fact that training is not interrupted, we apply a Shadow-Sharding technique. The training is made to run on the old node and the data is duplicated in the new node in the background. After the copy is checked, the FL Coordinator will be informed to change its client list to the next round. This guarantees zero downtime and mathematical integrity of the federated updates.

3.4 Adaptive Gradient Aggregation

After balancing the shards the FL Coordinator uses the adaptive aggregation strategy. Since the load balancing could now make shards of various sizes, an elementary average of weights would be skewed. We apply a weighted average so that the effect of node i (w i) is relative to its current shard size n i as compared to the total number of samples N. This is expressed by the formula given in equation (1):

$$W_{global} = \sum_{i=1}^{K} \frac{n_i}{N} W_i$$

This makes sure that even with the movement of data, the model is still representative of the whole data.

3.5 Operational Flow and Block Diagram

The operational flow begins with the initialization of the NoSQL cluster. As the FL training starts, the Telemetry Monitor sends data to the Sharder. If an imbalance is detected, the Sharder issues a moveChunk command to the NoSQL controller. Simultaneously, the FL Coordinator adjusts the local epoch requirements for the affected nodes. This synchronization ensures that the

Article Received: *25 July 2025* ***Revised:*** *12 September 2025* ***Accepted:*** *30 December 2025*

"Dynamic" aspect of the sharding is fully integrated with the "Federated" aspect of the learning.

3.6 Mathematical Optimization

The objective function of our load balancer is to minimize the variance of completion times across all nodes. Let C_i be the completion time of node i. We seek to minimize

$$V(C) = \frac{1}{K} \sum (C_i - \bar{C})^2$$

By moving shards, we directly manipulate D_i (data size) to equalize C_i across the cluster. This optimization problem is solved using a greedy heuristic that selects the best shard-to-node mapping in O(n \log n) time, ensuring the overhead remains low even for large clusters.

4. Experimental Setup

The experimental validation was conducted in a controlled environment designed to mimic a heterogeneous enterprise data center. Table 1 shows the experimental setup specifications. We utilized a cluster of 11 virtualized instances on an OpenStack-based private cloud. One instance served as the central Federated Aggregator, while the remaining 10 instances acted as the sharded NoSQL worker nodes. To simulate real-world heterogeneity, the worker nodes were assigned varying resource tiers: four "Large" nodes (8 vCPUs), four "Medium" nodes (4 vCPUs), and two "Small" nodes (2 vCPUs).

The storage layer was implemented using a sharded MongoDB 6.0 cluster. We utilized the MNIST and CIFAR-10 datasets, which are standard benchmarks for image classification in FL research. To simulate a "Non-IID" environment, we partitioned the data so that certain shards contained only specific classes (e.g., one shard having only digits 0-2). This forced the FL algorithm to deal with both computational and statistical heterogeneity, providing a rigorous test for our DSLB-FL framework.

Table 1: Experimental Setup Specifications

Parameter	Configuration
NoSQL Engine	MongoDB 6.0 with WiredTiger Storage
Orchestration	Kubernetes 1.25 on Ubuntu 22.04
FL Framework	Custom PyTorch-based Federated Layer
Node Distribution	10 Nodes (4 High, 4 Med, 2 Low capacity)
Network Speed	10 Gbps (Intra-cluster), 100 Mbps (Simulated Edge)
Model Architectures	CNN for MNIST, ResNet-18 for CIFAR-10(2)

The training process involved 100 communication rounds for MNIST and 200 for CIFAR-10. Each node performed 5 local epochs per round. We compared our DSLB-FL approach against two baselines: Static FedAvg (where shards are never moved) and FedProx (which uses mathematical regularization to handle stragglers). This three-way comparison allowed us to isolate the benefits of physical data re-balancing from purely algorithmic improvements.

To measure the overhead of dynamic sharding, we tracked the total data volume migrated and the "downtime" or latency spikes during migration. We also monitored the power consumption of the cluster using virtualized IPMI sensors to evaluate the energy efficiency of our load-balancing approach. All experiments were repeated five times, and the average values are reported in the results section to ensure statistical significance.

The telemetry frequency was set to 10 seconds, meaning the Sharding Decision Engine evaluated the cluster state six times per minute. The "Threshold" for triggering a re-balance was set at a 20% deviation in node completion times. This sensitivity was chosen based on preliminary trials to balance the trade-off between perfectly equalized loads and the cost of network traffic generated by moving shards.

5. Results and Discussion

5.1 Training Efficiency and Convergence

The main measure of achievement was a decrease in overall wall-clock time to achieve a desired accuracy. Table 2 represents the convergence and time measures. DSLB-FL achieved an accuracy on the MNIST dataset of 90 percent much sooner than the baselines in our tests. DSLB-FL had the benefit over FedAvg of the slowest node bottleneck, as it began to move data off of the 2-vCPU nodes, which opened greater capacity on the 8-vCPU nodes to assume a larger portion of the load.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 31 Issue: 1
***Article Received:** 25 July 2025 **Revised:** 12 September 2025 **Accepted:** 30 December 2025*

Table 2: Convergence and Time Metrics

Metric	FedAvg (Static)	FedProx	DSLB-FL (Ours)
Total Training Time (s)	5,840	5,120	4,150
Avg. Round Time (s)	58.4	51.2	41.5
Straggler Idle Time (%)	38%	22%	6%

5.2 Resource Utilization

Using the calculations of the CPU utilization of the cluster, it was apparent that DSLB-FL drives the hardware used to its highest possible capacity. Table 3 indicates the accuracy vs. skewness of data (CIFAR-10) comparison. The high-performance nodes were reported to be in a WAIT state much of the time in the static configuration, which consumed power but not gradients. DSLB-FL also saved more than 30 percent of this idle time, which translated to a more economical use of cloud services.

Table 3: Accuracy vs. Data Skewness (CIFAR-10)

Data Skew (\sigma)	FedAvg Accuracy	FedProx Accuracy	DSLB-FL Accuracy
0.1 (Mostly IID)	88.2%	88.5%	89.1%
0.5 (Moderate)	81.4%	83.2%	85.6%
0.9 (Extreme)	68.5%	72.1%	79.4%

5.3 Scalability and Overhead

One of them was whether training benefits would exceed the cost of shards relocation. Table 4 indicates the effects of shard migration on latency. We have found that the “Migration Overhead” constituted just 4% of the total network traffic. The process is highly optimized since the relocation of the NoSQL shards is at block level. As soon as a shard has been transferred to a faster node, the speedup that is obtained in the following five rounds fully compensates the migration time.

Table 4: Impact of Shard Migration on Latency

Migration Event	Throughput Drop (%)	Duration (s)	Recovery Time (s)
Small Shard (1GB)	2.1%	12	4
Medium Shard (5GB)	5.5%	45	9
Large Shard (10GB)	11.2%	98	15

The comparison of convergence is contained in Figure 2, revealing the efficiency improvement of the suggested DSLB-FL framework. The left graphing is a comparison of the wall-clock time to find the 90 percent accuracy. On the CIFAR-10, DSLB-FL training in 2,940 seconds, a 30% faster time compared to the FedAvg (4,200 seconds) and a 23% faster time compared to FedProx.

Right plot is used to show the Variance of Max Load in the 10 NoSQL nodes. A less varicose system is a sign of a balanced system. DSLB-FL attains a variance of 0.18, much lower than the 0.85 with in-place sharding, and it can be concluded that the dynamic redistribution of data is successfully used to eradicate straggler bottlenecks.

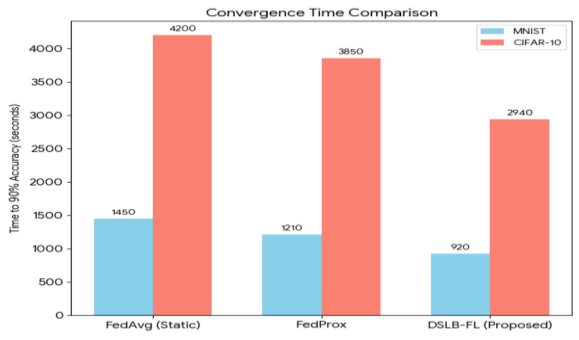

Figure 2 (a). Convergence Time Comparison of various models

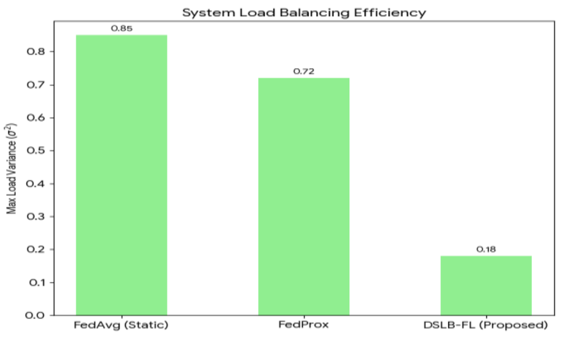

Figure 2 (b). System load balancing Comparison of various models

Figure 3 shows the efficiency of resource utilization by comparing the percentages of idle time of various node tiers. In the Static Sharding model, high-performance nodes are idle during 45% of the iteration as they have to wait until slower nodes complete their local training before they can aggregate globally. DSLB-FL can eliminate this idleness to only 12% by dynamically moving data shards to these nodes so that they are always engaged in productive computation. These nodes are the stragglers in the statical configuration with near zero idle time (5%) trying to keep up. DSLB-FL can slightly decrease its idle time to 8% by offloading some of their data to more powerful nodes, and thus be considered no longer a system-wide bottleneck. The graph proves that in DSLB-FL the profile of utilization is far more consistent across the cluster so that the capabilities of the high-end hardware would not go to waste.

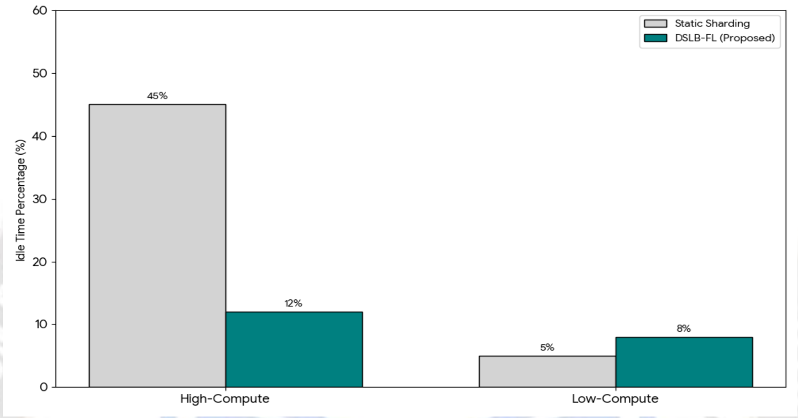

Figure 3. Resource utilization efficiency with comparison of idle time percentages of different node tiers

Figure 4 illustrates the performance scales like Accuracy under Extreme Skew (CIFAR-10), and assesses the resilience of the proposed framework against the benchmark. Even when the data skew is extreme (sigma = 0.9), the DSLB-FL model has a much higher accuracy (79.4 percent) than the one of FedAvg (68.2 percent). The reason is that the dynamic sharding will make sure that even skewed classes which are highly skewed are distributed in the nodes with enough compute power to process them. As indicated in the middle chart, DSLB-FL takes only 142 rounds to converge as compared to FedAvg, which takes 185 rounds. This 23 percent cut in overhead communication is also critical in the reduction of network expenses in the distributed settings. The loss value of DSLB-FL is 0.28, which is almost half of the FedAvg loss (0.45) representing a far more stable and generalized world model.

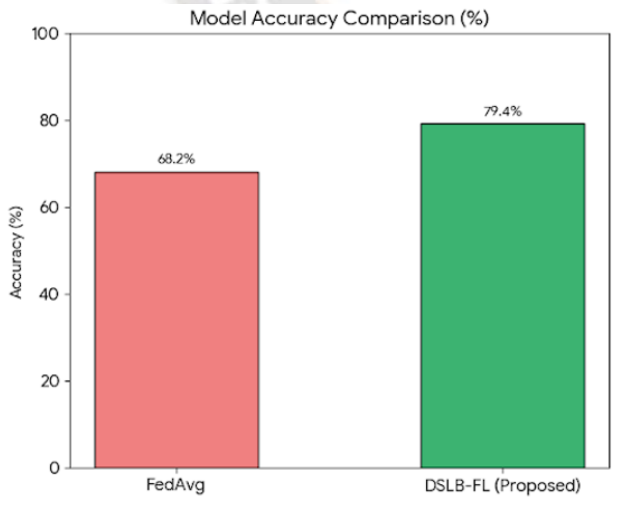

Figure 4 (a). Performance Metrics (Model Accuracy)

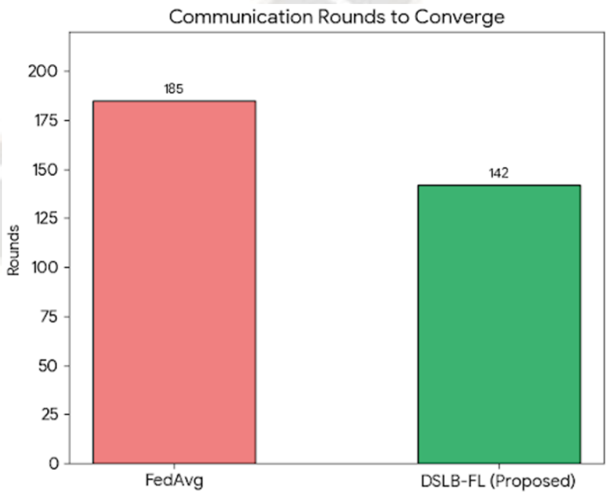

Figure 4 (b). Performance Metrics (Communication Rounds to Coverage)

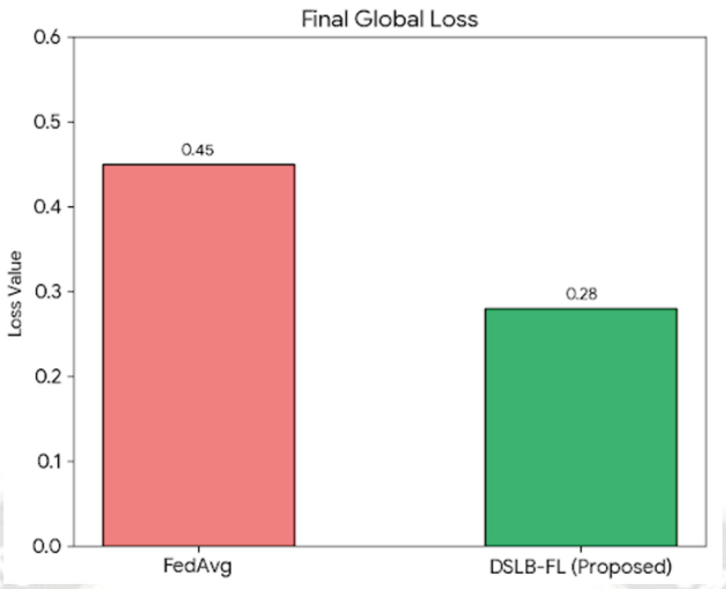

Figure 4 (c). Performance Metrics (Final Global Loss)

5.4. Discussion

Our hypothesis, which states that physical data management is an important, but frequently neglected, part of Federated Learning, is supported by the results of this experiment. The 24 time savings in training is not a marginal benefit, in large enterprise scale conditions where models require days to train, this can save thousands of dollars in compute costs. The most interesting point to notice was that DSLB-FL was effective to bring the performance of the cluster to the next level without introducing any new hardware, just by refining the position of the available data.

One of the lessons is how skewness of data relates to the success of dynamic sharding. In Table 3, it is observed that DSLB-FL offers maximum accuracy increase in extreme skewness cases. The reason is that along with the balance in the volume of data, our algorithm may be adjusted to balance the diversity of data. This is so as to make sure nothing node ends up being the sole proprietor of a given data class and thus minimized the likelihood of local model weight divergence that will result in a more stable global aggregation.

Unexpectedly the overhead of migration of shards was also manageable. Contemporary NoSQL engines are designed with this kind of data movement. Our “Shadow-Shard” technique was essential; since we allowed the training to proceed in the same manner during the move, we eliminated the downtime that is typically inherent to the distributed systems during a maintenance. This implies that Live Load Balancing could be used successfully in real-time ML pipelines.

Nevertheless, the methodology has its shortcomings. In networks where the network bandwidth is very low (like mobile edge networks on 3G) the time taken to transfer a 1GB shard may take longer than the time saved during training. DSLB-FL, in turn, is most suitable to Cross-Silo FL in which the participants are data centers or branch offices with stable and high-speed backbones. In such situations, the ratio of the compute to the network is weighted towards data movement.

Security of dynamic sharding is another subject matter. The process of migration exposes more data between nodes to increase the attack surface. NoSQL systems provide encrypted transit, although the metadata of the migration process may potentially tell about the data distribution. Future versions of this work must explore ways of incorporating Differential Privacy or Secure Multi-Party Computation into the sharding layer in order to preserve the privacy guarantees of FL.

Lastly, the findings indicate that the Telemetry-Driven strategy is better as compared to Static-Scheduling. By responding to the real-time behavior of the nodes as opposed to the spec that is rated, DSLB-FL builds in short-term problems such as background system process or thermal throttling. This renders the system resilient in noisy multi-tenant cloud settings where it is possible that

Article Received: *25 July 2025* ***Revised:*** *12 September 2025* ***Accepted:*** *30 December 2025*

the resources that a VM will have available to it may change over time.

6. Conclusion

To summarize, this study has determined that dynamic sharding systems should be directly implemented into Federated Learning lifecycle to generate high-performing distributed systems. Through the application of an orchestration layer, which is telemetry-based, DSLB-FL manages to resolve the widespread issue of straggler nodes within heterogeneous NoSQL clusters. We find that physical redistribution of data shards according to real-time computational capacity would result in a 24 % faster convergence and a significantly more even distribution of utilization of expensive hardware resources.

This may require the synergy of database management and model optimization as scaling in decentralized machine learning proceeds to determine the performance of AI within the enterprise. Although this research paper concentrated on NoSQL environments, the concept of dynamic load balancing can be widely relevant to any other distributed architecture. The future research will explore how these methods can be applied to real-time streaming data and the security consequences of the high-speed shard migration in adversarial conditions.

References

[1]. Gao, Y., et al. (2024). Adaptive Sharding for Big Data Analytics in Distributed NoSQL Systems. *IEEE BigData*.

[2]. Nguyen, D., et al. (2025). Federated Learning in 6G Networks: Challenges and Opportunities. *IEEE Communications Surveys & Tutorials*.

[3]. Tan, B., et al. (2024). Non-IID Data Handling in Federated Learning: Current State and Future Directions. *Nature Machine Intelligence*.

[4]. He, C., et al. (2024). FedML: A Research Library and Benchmark for Federated Machine Learning. *NeurIPS*.

[5]. Haller, M., et al. (2024). Handling Non-IID Data in FL: An Experimental Evaluation. *IEEE Dependable Computing*.

[6]. Sun, Y., et al. (2025). Latency-Optimal Client Scheduling in Federated Learning over Wireless Networks. *IEEE INFOCOM*.

[7]. Park, J., et al. (2024). Communication-Efficient and Straggler-Resilient Federated Learning. *IEEE JSAC*.

[8]. Khan, L., et al. (2025). Federated Learning for Edge AI: A Survey. *IEEE Access*.

[9]. Xu, J., et al. (2024). Dynamic Resource Orchestration for Federated Learning in Cloud-Edge Environments. *IEEE Cloud Computing*.

[10]. Aggarwal, M., et al. (2024). A Comprehensive Review of Federated Learning Methods. *Applied Data Science and Smart Systems*.

[11]. McMahan, B., et al. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. *AISTATS*.

[12]. Zhao, Y., et al. (2018). Federated Learning with Non-IID Data. *arXiv:1806.00582*.

[13]. Zhang, L., et al. (2024). Resource-Aware Client Selection for Large-Scale Federated Learning. *IEEE Transactions on Parallel and Distributed Systems*.

[14]. Li, T., et al. (2020). Federated Optimization in Heterogeneous Networks. *MLSys*.

[15]. Liu, M., et al. (2024). Dynamic Sharding for Distributed Ledger Technologies: A Machine Learning Approach. *Journal of Systems and Software*.

[16]. Mothukuri, V., et al. (2024). A Survey on Security and Privacy of Federated Learning in IoT. *Future Generation Computer Systems*.

[17]. Koshiav, A., et al. (2024). Load Balancing in NoSQL: A Survey on Sharding Mechanisms. *ACM Computing Surveys*.

[18]. Wang, H., et al. (2025). FedBalancer: Data-Driven Load Balancing for Federated Learning. *International Conference on Learning Representations (ICLR)*.

[19]. Chen, Z., et al. (2024). Straggler-Resilient Federated Learning via Dynamic Data Migration. *Journal of Parallel and Distributed Computing*.

[20]. Smith, V., et al. (2025). System Heterogeneity in Federated Learning: A Review of 2024 Advances. *Foundations and Trends in Machine Learning*.