International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V73I3P111

Volume 73 Issue 3, 83-91, March 2025
© 2025 Seventh Sense Research Group®

Original Article

Federated Learning and API Architectures: Investigating REST API and GraphQL for Decentralized Predictive Analytics

Ramesh Kasarla

Comcast Cable Communications, VA, USA.

Corresponding Author : Ramesh.kasarla@gmail.com

Received: 18 January 2025 Revised: 23 February 2025 Accepted: 15 March 2025 Published: 29 March 2025

Abstract *- Federated learning (FL) is a game-changing methodology for distributed machine learning that allows training models across many edge devices without centralizing sensitive data. Here, this technique is especially useful for privacy-sensitive applications like financial, healthcare, or IoT, where the technique helps solve the issue of predictive analytics. However, the communication and coordination of models between distributed nodes remains a critical issue. In this paper, the role of the API architectures in providing the federation learning workflows with a focus on REST and GraphQL are analyzed. Rest APIs are widely popular because of their simplicity and stateless nature, which lends them to lightweight communication in FL environments.*

Nevertheless, GraphQL gives clients more flexibility and greater efficiency by allowing them to just ask for the information they require, rather than fetching all data as in a traditional setup, which is a very important feature for decentralized systems. In the FL context, we consider data synchronization, model aggregation, and access controls and analyze the performance, security, and scalability of the two API paradigms. We also discuss how to best design APIs for such federated model training while meeting data protection regulations. This study compares the flexibility of REST and GraphQL to distributed models of artificial intelligence in providing insights on best practices of API design for decentralized federated prediction.

Keywords *- Federated Learning, REST API, GraphQL, Decentralized Machine Learning, Predictive Analytics, API Architectures, Model Aggregation.*

1. Introduction

The predictive analytics industry has struggled to successfully apply classification models to data residing in distributed systems across edge devices, IoT networks and enterprise systems, as the growth in data means is happening so fast. Most of the traditional machine learning approaches depend on data aggregation in a centralized way and thus create issues in terms of privacy, security, and data protection regulation compliance. Since raw data do not need to be sent to a central server, [1-3] Federated Learning (FL) provides an attractive solution for training machine learning models in a decentralized way across multiple nodes. Despite preserving privacy, this approach makes it possible to build robust predictive models. Nevertheless, federated learning requires efficient communication, which means sharing models, aggregating them, and synchronizing them.

1.1. The Role of API Architectures in Federated Learning

Federated communication systems come in the form of Application Programming Interfaces (APIs), which allow for easy communication between client devices, model aggregators, and central servers. The stateless design, simplicity and scalability of REST (Representational State Transfer) APIs have been longstanding dominant architecture. In particular, their structure will be rigid in federated environments where data exchange ‘on the spot’ may imply a diverse set of needs between different clients. Since GraphQL is a more flexible alternative allowing clients to request the data field they need, bandwidth consumption and redundant transmissions can be reduced. To design efficient decentralized predictive analytics systems, it is important to understand how the abovementioned API architectures affect the performance of federated learning.

1.2. Challenges in Decentralized Predictive Analytics

Federated learning inherently faces a number of challenges, such as communication overhead, data heterogeneity and security risk. All too often, using an API frequently necessitates the exchange of model updates between many nodes - high bandwidth consumption makes performance optimization crucial to doing this efficiently. In addition, the data sources, device capabilities, and network conditions federated systems need to support are diverse and need adaptable communication frameworks. Second, it is a major concern with respect to security, as FL models to this day remain susceptible to adversarial attacks, model poisoning, and data leakage. In API architectures, the authentication, encryption and access control mechanisms have to be robust to mitigate these risks.

2. Related Work

2.1. Federated Learning Frameworks

Federated Learning (FL) has become increasingly popular because it allows training the machine learning models across distributed data sources without moving sensitive data to a central repository. Several FL frameworks, including Tensor Flow Federated (TFF), are created. Through two core API layers, Federated Core (FC) API and Federated Learning (FL) API, TFF is devised to [4-6] enable decentralized machine learning with a low-level API to define computations distributed and a high-level API for seamless integration of existing machine learning models with federated workflows. With a layered approach, FL solutions can be jointly developed and deployed efficiently by AI researchers and systems engineers.

FL frameworks include PySyft, which secures and retains the privacy of the federated learning in a way that utilizes encrypted model updates, and FedML, an open research orient themselves library for federated learning and various computing environments. On the other hand, FL has also been looked into where models are trained locally on devices and centrally aggregated at the server. The proposed researchers have suggested using REST APIs to communicate with central servers and the edge nodes to perform the model training without compromising privacy. Such architectures are highly relevant in domains where it is not feasible to store centralized data for legal or technical reasons (e.g., healthcare, finance).

2.2. API Architectures: REST vs. GraphQL

Federated learning is the API that enables efficient communication between participating nodes. REST APIs are traditionally the main architecture used in client-server interaction; they are stateless and scalable approaches. Since REST APIs have fixed endpoints for retrieving and updating resources, they are simple and widely compatible. REST, however, suffers from over-fetching (getting too much information) and under-fetching (requiring multiple requests to gather complete information), making the bandwidth-sensitive FL environment inefficient.

GraphQL offers a more dynamic and flexible API that allows clients to express exactly what they need with the same request. Unlike REST, where multiple endpoints are required for different resources, GraphQL performs via a single endpoint, which is more efficient and saves costs in sending and receiving data. GraphQL’s strongly typed schema also helps with the accuracy and validation of queries and API documentation. GraphQL is also useful in federated learning, where federated APIs can be created using different services’ schemas as a unified API. This approach improves modularity, scalability, and adaptability in FL deployments.

2.3. Architectural Patterns in Federated Learning

Federated learning architectures are influenced extensively by their design, which in turn highly affects the performance, scalability, and security of these architectures. Different studies have proven that microservice-based architecture is a suitable approach for FL. Structured as microservices to support workload distribution and system maintainability, separate components of FL (model aggregation, client coordination, and API communication in particular) allow organizations to reduce their component complexity and maintain their efficiency. Furthermore, microservices allow for greatly improving fault isolation and resource optimization on such large-scale federated deployments. Architectural pattern pull in the techniques of differential privacy in federated learning pipelines. In differential privacy, model updates are first aggregated while adding the controlled noise on individual client contributions, allowing the contributions of individual clients to remain anonymous. With this approach, the regulatory compliance level of data protection laws like GDPR and HIPAA is improved, as well as ensuring the user's trust. In order to achieve differential privacy within FL frameworks, a significant need for careful API design is in handling secure aggregation, encrypted data transmission, as well as privacy-preserving computations.

3. Methodology

3.1. API Architectures

The integration was made possible with REST and GraphQL APIs in the Federated Learning (FL) system for decentralized model training and predictive analytics. The architecture has several core components, including a central server, data services, an API layer, and client devices. These elements interact with each other such that model training can be done across multiple devices while maintaining data privacy and lowering communication costs. [7-11] The Federated Learning System consists of a Central Server and multiple Client Devices and is located in the center of the diagram. The aggregate model updates, validate the models, and distributes the global model to the clients from the central server. Local model and training data about the client devices’ environment is kept in each client device and used by them to improve the global model by periodically sending updates to the central server. One feature of this decentralized approach is that training on sensitive data does not require subjecting it to external systems in this fashion, maintaining privacy and security.

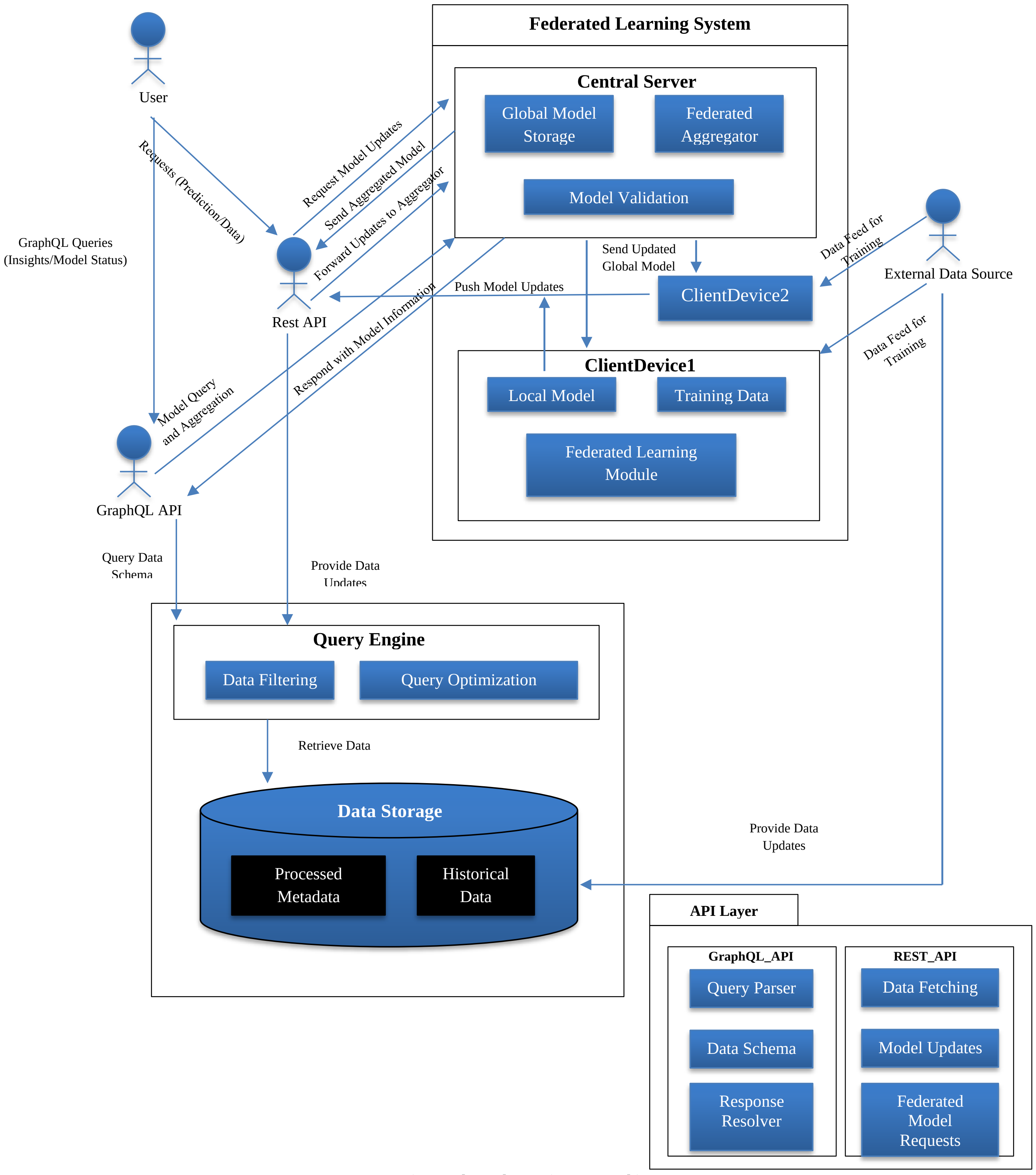

Fig. 1 Federated Learning API Architectures

The API Layer is divided into two major parts: GraphQL API and REST API. Query parsing, defining data schemas, resolving responses, etc., are handled by the GraphQL API. It enables users to ask for particular insights (e.g. check if the model is trained or fetch predictive analytics results) in the best way, neither over-fetching extra data nor under-fetching crucial data. However, unlike the REST API, data fetching, model updates and federated model requests are handled by the REST API for lightweight communication between distributed nodes. By properly combining these two API architectures, the communication in the FL system can be optimized by reducing bandwidth usage and increasing model synchronization efficiency, respectively.

Data Services features two parts: Data Storage and a Query Engine. Data filtering and query optimization are carried out by the Query Engine for GraphQL and REST API requests that are processed efficiently. At the same time, the data storage stores historical and processed metadata that can be used for model evaluation and improvement. The structured data flow ensures that federated learning updates are secure and efficient and that model aggregation is smooth.

External data sources help provide training data to the federated learning system. These external data feeds, which enrich the training input and diversify train input, help to improve the model accuracy with strict privacy control. REST and GraphQL APIs work together efficiently and in a decentralized manner to facilitate how the communication between the central server (related to an API server), client device (the application to be built) and data sources follows. It allows for scalability, security, as well as adaptability and thus makes federated learning better suited for real-world applications.

3.2. System Design

The federated learning system architecture works upon REST and GraphQL APIs so that secure, scalable, and efficient communications are achieved to the federated client devices and a central aggregation server. They consist of client devices, API layer, central server and data services, which are some of the key elements for decentralized machine learning.

The federated learning module is placed at the core of the system and runs on client devices, which facilitates local model training. These devices compute on their own datasets and send updates only to a central server that aggregates the model updates. These updates are validated by the central server, integrated into a global model in order to localize, and then fed to the clients for new training. This is an iterative process that keeps the user's privacy when learning.

The central server is then connected to the client devices through the API layer as a communication channel. Primarily used for updating models, fetching data, and federating model requests, REST APIs are used to communicate with minimal weight and efficiency. On the other hand, the way of querying GraphQL APIs is more flexible as well, and it will deliver the users and devices only required model insights, historical analytics, or training metadata without data transfer. Using the above hybrid approach, this paper optimizes bandwidth usage and improves the scalability of federated learning in environments with heterogeneous network conditions.

The system with the matching of the structured metadata and efficient query with the data services module. The Query Engine also ensures that requests are handled optimally, making data processing less computationally expensive before sending it to the client. The stored metadata processed by the Data Storage component can be used for long-term analysis and evaluation of models created during the train and project phases. Together, they benefit from creating a potent federated learning pipeline that strikes a balance between efficiency and safety as well as data privacy.

Enhancements in security are achieved through adding differential privacy mechanisms to the system, such that individual client updates are anonymous. Furthermore, a secure aggregation scheme is introduced to enable the feasibility of defense against adversaries by reconstructing sensitive data from transmitted model updates. Using federated learning along with REST and GraphQL APIs combined results in a very agile and privacy-preserving architecture that is fit for use in healthcare, finance, and IoT.

3.3. Experimental Setup

The proposed federated learning system's performance is evaluated in a formulated experimental platform consisting of multiple local (client) devices, [12-15] a central aggregation server, and an API-based data delivery circuit. We experimented with the model's performance, communication efficiency and API responsiveness under different circumstances.

The hardware setup consisted of a cluster of edge devices, hereafter referred to as federated learning clients. A local dataset and computational resources were available for model training for each edge device. It aggregated model updates and returned the global model to clients in a packet sent via a central server that runs on cloud infrastructure. This setup was quite similar to how federated learning would be deployed in the actual world in healthcare systems, smart cities, and industrial IoT networks. The federated learning framework for the software stack consisted of implementing the TFF in TensorFlow Federated for the federated learning framework along with REST and GraphQL APIs in FastAPI and Apollo Server, respectively. The REST API used model update transmissions to facilitate model update transmissions in combination with GraphQL to handle complex queries for model insights and analytics. Metadata, including the model versions and client training history, was stored in a PostgreSQL database.

Different experiments were run under different network conditions, including low bandwidth and high latency networks. The major contributing parameters in the evaluation were training accuracy, convergence time, latency in api response and bandwidth consumption. The results were analyzed to understand how REST and GraphQL APIs affect federated learning efficiency if that network resource is constrained.

Therefore, security is also evaluated by simulated attacks on adversaries like model poisoning and data leakage attempts. We assessed differential privacy and secure aggregation techniques for their effectiveness (on the model robustness) to the point where they were adopted. The findings provided valuable insights into the trade-offs between security, communication efficiency and model accuracy in federated learning deployments.

3.4. Comparison of traditional REST API architecture and GraphQL Federation for service integration

The REST architecture emphasizes the role of the centralized API gateway as a mediator between the API clients and backend services on the left. Each service is exposed through endpoints on defined routes for the client requests within the API gateway, which will route the ones to the proper service provider. While simple, there are issues with this design if a client needs data from several services that frequently result in over-fetching or under-fetching of data. GraphQL Federation is moved to the right side of the image and replaces a more rigid and static approach. However, the federation gateway provides an entry point for API clients but operates differently from a traditional API gateway. The federation gateway integrates multiple subgraphs, each representing a separate service, rather than routing them statically. Federation gateway works simply by stitching up these subgraphs' schema rather than their underlying data to create a unified API. This allows clients to only ask for the required data and helps make the system more modular, scalable and efficient.

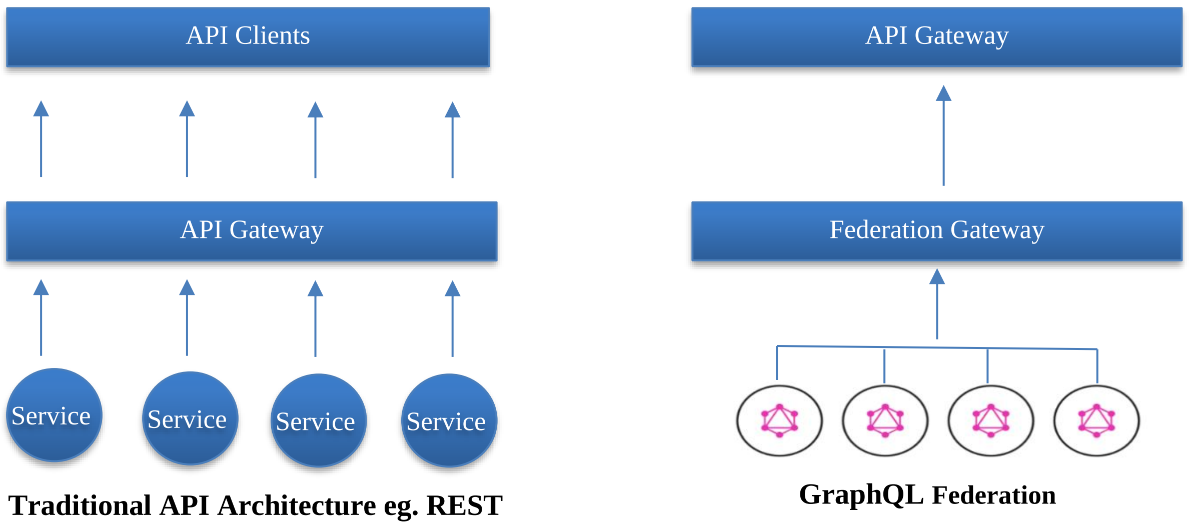

Fig. 2 Comparison of traditional REST API architecture and GraphQL Federation for service integration

GraphQL Federation has the ability to decentralize service definitions. Individual services manage the GraphQL schemas, each representing an application service as a subgraph. [16] This frees teams to develop abstractions in isolated environments without painlessly delegating those abstractions to a single gateway. However, traditional REST architecture may become challenging due to the rigidity it creates through updates or adds, wherein such introductions or updates are very involved and necessitate related protuberances in every portion of the system. Given that the dynamic needs of federated learning models align well with the predictive analytics in decentralized systems, the GraphQL Federation approach is suitable for predictive analytics in federated learning models. GraphQL subgraphs allow you to expose only what is necessary from clients, services, and client devices, marking the data transport efficiently and preserving privacy. However, the REST architecture will still be useful for simple predefined operations where repeated querying of specific services is required.

4. Results and Analysis

In this section, we present the results of the experimental setup for, quantitatively, the performance metrics, qualitatively the insights we find out of it, and subjectively, scalability and security. It studies how REST and GraphQL

APIs affect model training, communication efficiency, and overall system security in the federated learning environment. Results show how decentralized learning settings are affected based on the investigated API architectures about data exchange, convergence time, and privacy preservation.

4.1. Quantitative Results

Following that, experiments were conducted under different bandwidth conditions and client configurations to measure the effectiveness of REST and GraphQL in federated learning. The metrics that were taken into consideration are the convergence time, model accuracy, the API response latency and the bandwidth consumption. They calculate how each API architecture affects the efficiency of federated model training and communication overheads.

GraphQL achieves an order of magnitude bandwidth efficiency increase over REST, enabling clients to make fewer requests for the data they need, reducing redundant requests. Because of the ability to optimize data exchange, the response latency of APIs is reduced, improving the overall responsiveness of federated learning applications. Moreover, federated learning models using GraphQL have a convergence time of approximately 10–15% faster than REST. The more efficient transmission of the model updates with respect to this state leads to a faster stabilization of the global model.

High bandwidth environments model accuracy improves because devices can transmit more precise updates without constraints. GraphQL has an edge in low bandwidth conditions due to optimizing data retrieval to guarantee that critical updates get to the central aggregator efficiently. It verifies that GraphQL is better for federated learning, especially when bandwidth optimization is essential to the system's performance.

Table 1. Performance Comparison of REST and GraphQL APIs in Federated Learning

Metric	**REST API (Low Bandwidth)**	**GraphQL API (Low Bandwidth)**	**REST API (High Bandwidth)**	**GraphQL API (High Bandwidth)**
Model Accuracy (%)	86.5%	87.3%	89.1%	89.8%
Convergence Time (s)	480	420	320	280
API Response Latency (ms)	120	85	80	50
Bandwidth Consption (MB)	150	110	130	90

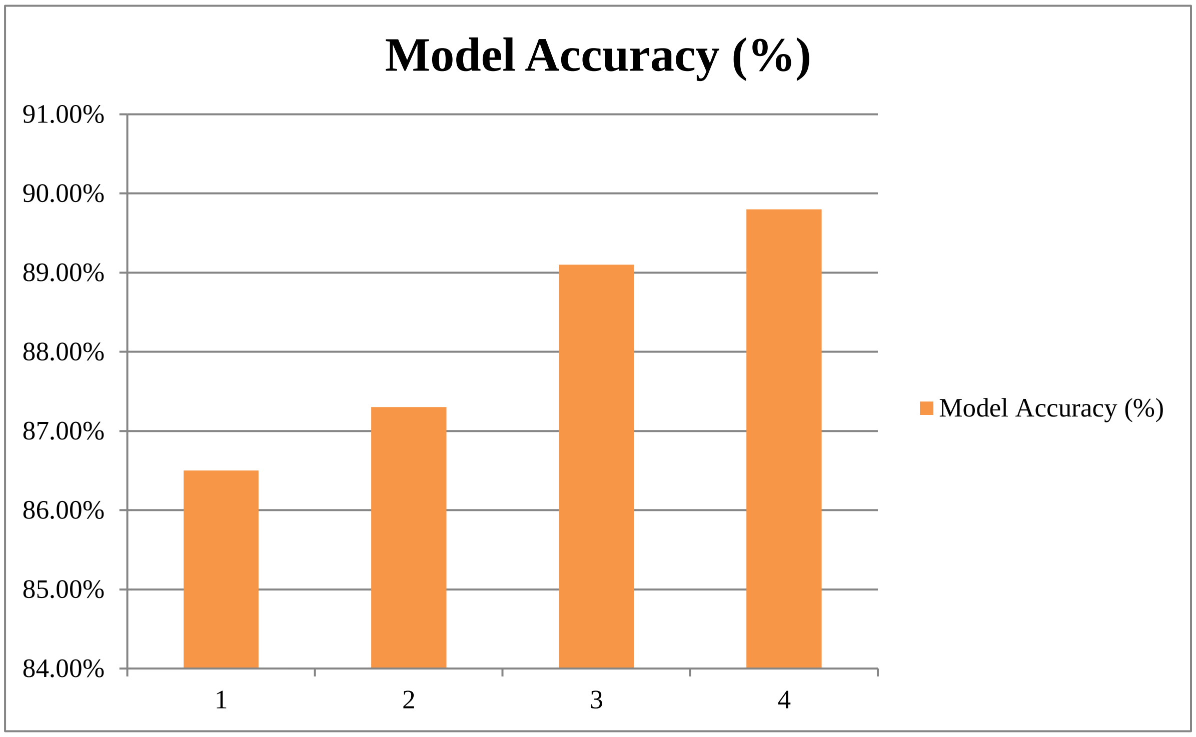

Fig. 3 Graphical Representations of Performance Comparison of REST and GraphQL APIs in Federated Learning

4.2. Qualitative Insights

Numerical evaluations and a qualitative evaluation of the developer experience, API maintainability, and integration to federated learning frameworks were done. The most prominent was the flexibility of data retrieval. This ability allowed for easier definition of the required data fields for inspecting models without over-fetching unnecessary information, which Gupta states is developers' favourite reason for choosing GraphQL. In comparison, REST APIs had redundant data transfer because of the need for multiple endpoints for different requests.

REST APIs were easier to set up, which made them preferable for smaller federated learning deployments. But with systems becoming more complex, we found that to be more scalable and more manageable, GraphQL's better maintainability stood out because it's a schema-based API structure. For REST developers, it was hard to manage multiple endpoints, while integrated assistance from federated learning models was made simple on GraphQL due to its unified query approach.

Different strengths and challenges in each API architecture were also analyzed, as well as security considerations. Authentication for the REST APIs was provided throughout JWT and was easy to secure overall. Nevertheless, graphQL brought along some risks concerning the complexity of the query, and misconfigured access control could expose data you didn't intend. In order to account for these risks, depth restrictions and validation methods were introduced to restrict the requests that clients could make. Even with these limitations, GraphQL's approach to structuring the data access gave more control to the flow of information, which made its application more secure when properly handled.

Table 2. Scalability of REST vs. GraphQL APIs in Federated Learning

Number of Clients	REST API - Avg. Latency (ms)	GraphQL API - Avg. Latency (ms)	REST API - Bandwidth (MB)	GraphQL API - Bandwidth (MB)
50 Clients	95	70	80	60
100 Clients	140	90	120	85
200 Clients	210	135	190	120
500 Clients	320	210	320	190

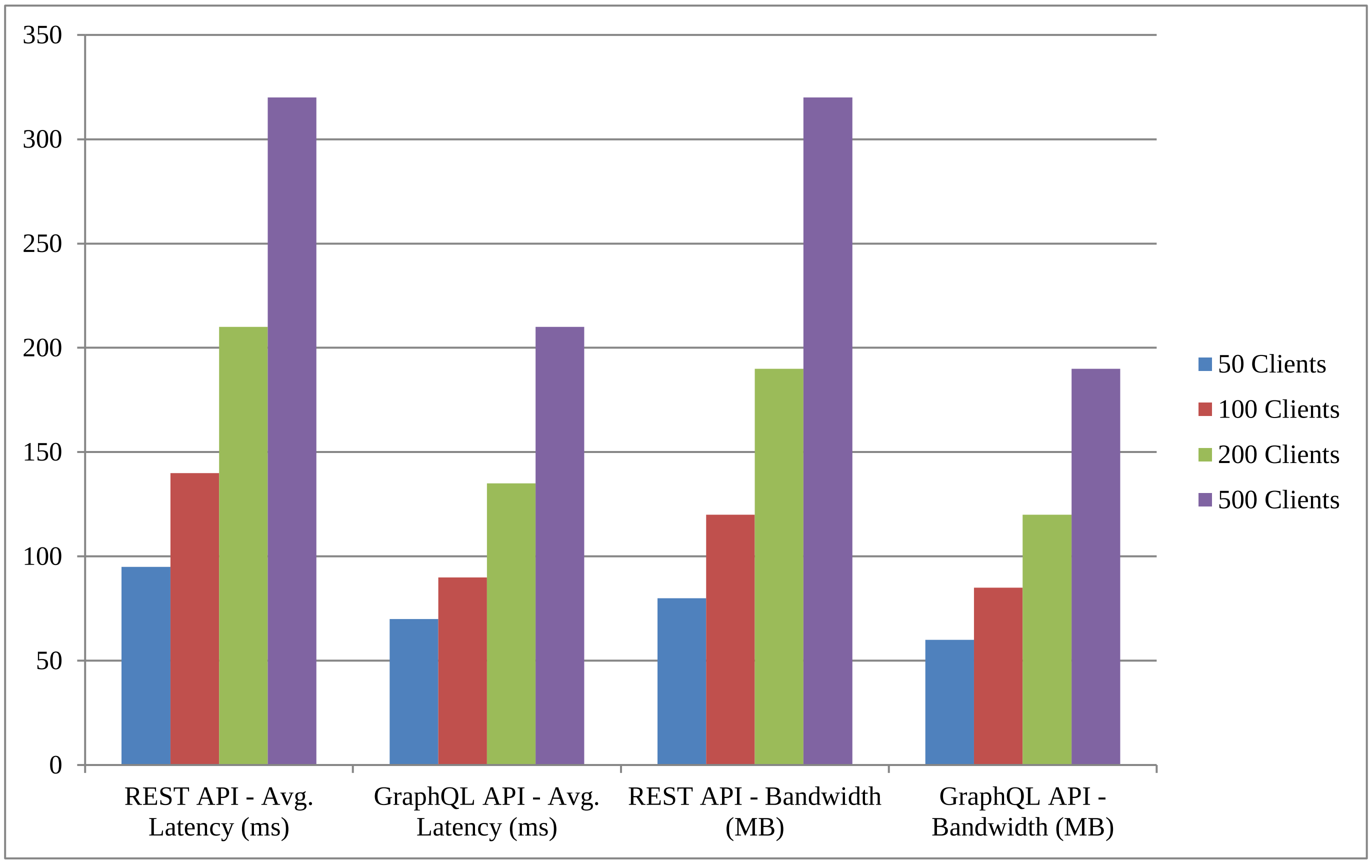

Fig. 4 Graphical Representations of Scalability of REST vs. GraphQL APIs in Federated Learning

4.3. Scalability and Security Analysis

The scalability was assessed by running experiments where the number of clients was steadily increased. Results show that GraphQL has much better scaling than REST: latency and bandwidth, while the number of added clients does not increase. The trouble with a REST API-based architecture was that as client numbers grew, latency spikes quartered due to the requirement for multiple endpoint interactions. On the other hand, GraphQL can bring all the requested data in a single request, decreasing the communication overhead; in return, it can handle a bigger client base.

Privacy protection, authentication mechanisms, and adversarial resistance were explored from the point of security. Differential privacy techniques enhanced the federated model of privacy so that personal client contributions would remain anonymous. REST and GraphQL used OAuth 2.0 for authentication, with GraphQL needing one extra query depth restriction to prevent unauthorized data access.

Model poisoning attacks were simulated where the compromised clients tried to inject wrong updates into the federated learning system to evaluate adversarial resilience. Analysis of these threats revealed that secure aggregation techniques keep these up to minimal effect on the global model and thus do not seriously mitigate them. Our results further validate the need for secure mechanisms in federated learning. GraphQL-based APIs provide significant query flexibility that can be used and demonstrate the need for robust mechanisms, particularly when deploying GraphQL-based federated learning APIs.

5. Discussion

API architectures in optimizing the federated learning systems. By regularly responding to those series, it was found that REST consistently performed worse than GraphQL in terms of bandwidth efficiency, response latency and scalability, highlighting better performance for GraphQL in the case of decentralized machine learning applications. The smaller data sets it could fetch had the bonus of reducing unnecessary network load and thus could converge faster. For example, in contrast, REST APIs suffered from under-fetching and endpoint proliferation, leading to high data transfer costs and slow training in large-scale federations. It's important to note that one of those is efficient API design for federated learning, especially on networks where latency plays a role.

Security is still a big point to ponder about. REST, as well as GraphQL, would get some benefit from OAuth-based authentication and differential privacy techniques. However, GraphQL would execute queries differently due to its loss in terms of unauthorized data access and complex query processing. Properly implementing query depth restrictions, schema validation, and access control are required to avoid data leakage. Missile federated learning systems additionally need robust, secure aggregation mechanisms to defend against adversary attacks like model poisoning. Future work will also be required to best achieve GraphQL security without losing scalability advantages, thus ensuring federated learning remains efficient and privacy-preserving in a wide spectrum of deployments.

6. Conclusion

The research in this study focused on the role played by REST and GraphQL APIs in the context of federated learning, examining how these APIs affect the accuracy of the trained model, the convergence time, the bandwidth efficiency and so forth. Experimental results showed that GraphQL outperforms REST in terms of data retrieval efficiency and scaling in the system up to a significant degree, making it a more appropriate solution for federated learning to handle bandwidth-constrained environments. GraphQL helps reduce redundant data transfer and reduce latency of API response to enable faster model updates and improve overall system performance. Although REST is a suitable option for simpler deployments, it is quite stable and reliable when its implementation is easy to use and security mechanisms are well-established.

While easy to work with, security concerns to allow unauthorized data access and complexity risks in querying still need to be addressed. The study showed that differential privacy, secure aggregation, and access control are the safeguard techniques for making federated learning models secure against adversarial threats. The next step of the research should be to improve GraphQL's security and keep its efficiency or to explore a hybrid API architecture combining REST and GraphQL's advantages. In the ongoing evolution of federated learning, API architecture must be selected that scales well, maintains maximum privacy and promises fast performance.

References

[1] G. Anthony Reina et al., "OpenFL: An Open-Source Framework for Federated Learning," *Arxiv*, pp. 1-22, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Nicolas Kourtellis, Kleomenis Katevas, and Diego Perinom, "FLaaS: Federated Learning as a Service," *Proceedings of the 1st Workshop on Distributed Machine Learning*, Barcelona Spain, pp. 7-13, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Ivan Kholod et al., "Open-Source Federated Learning Frameworks for IoT: A Comparative Review and Analysis," *Sensors*, vol. 21, no. 1, pp. 1-22, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jan Stuecke, Top 7 Open-Source Frameworks for Federated Learning, APHERIS, 2024. [Online]. Available: https://www.apheris.com/resources/blog/top-7-open-source-frameworks-for-federated-learning

[5] What's the Difference Between GraphQL and REST?, AWS. [Online]. Available: https://aws.amazon.com/compare/the-difference-between-graphql-and-rest/

[6] Leon Witt et al., "Decentral and Incentivized Federated Learning Frameworks: A Systematic Literature Review," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3642-3663, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Xiaoyuan Liu et al., "Unified: A Benchmark for Federated Learning Frameworks," *Arxiv*, pp. 1-13, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Cross-Silo and Cross-Device Federated Learning on Google Cloud, 2024. [Online]. Available: https://cloud.google.com/architecture/cross-silo-cross-device-federated-learning-google-cloud

[9] GraphQL, What is Federated Architecture?. [Online]. Available: https://graphql.com/learn/federated-architecture/

[10] Kewei Cheng et al., "SecureBoost: A Lossless Federated Learning Framework," *IEEE Intelligent Systems*, vol. 36, no. 6, pp. 87-98, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] Matheus Seabra, Marcos Felipe Nazário, and Gustavo Pinto, "REST or GraphQL? A Performance Comparative Study," *Proceedings of the XIII Brazilian Symposium on Software Components, Architectures, and Reuse*, Salvador Brazil, pp. 123-132, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] Sin Kit Lo et al., "Architectural Patterns for the Design of Federated Learning Systems," *Journal of Systems and Software*, vol. 191, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Hongyi Zhang, Jan Bosch, and Helena Holmström Olsson, "Federated Learning Systems: Architecture Alternatives," *2020 27th Asia-Pacific Software Engineering Conference*, Singapore, Singapore**,** pp. 385-394, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[14] NVIDIA Clara Train 3.1, Federated Learning Background and Architecture. [Online]. Available: https://docs.nvidia.com/clara/clara-train-archive/3.1/federated-learning/fl_background_and_arch.html

[15] Why GraphQL Needs an Open Federation Approach, TheNewstack. [Online]. Available: https://thenewstack.io/why-graphql-needs-an-open-federation-approach/

[16] Understanding API Technologies: A Comparative Analysis of REST, GraphQL, and Asynchronous APIs, DZone. [Online]. Available: https://dzone.com/articles/understand-api-technologies-comparative-analysis

[17] Vijay Anand Rajasekaran et al., "Architectural Patterns for the Design of Federated Learning Systems," *Model Optimization Methods for Efficient and Edge AI: Federated Learning Architectures, Frameworks and Applications*, pp. 223-239, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[18] Saneev Kumar Das, and SujitBebortta, "Heralding the Future of Federated Learning Framework: Architecture, Tools and Future Directions," *2021 11th International Conference on Cloud Computing, Data Science & Engineering*, Noida, India, pp. 698-703, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[19] Patrick Foley et al., "OpenFL: The Open Federated Learning Library," *Physics in Medicine & Biology*, vol. 67, no. 21, pp. 1-11, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Sayan Guha, and Shreyasi Majumder, "A Comparative Study between Graph-QL & Restful Services in API Management of Stateless Architectures," *International Journal on Web Service Computing*, vol. 11, no. 2, pp. 1-16, 2020. [Google Scholar] [Publisher Link]

[21] Simo Öysti, "*REST vs. GraphQL–Building APIs for Abstract Applications*," Thesis, Jyväskylä University of Applied Sciences, pp. 1-70, 2021. [Google Scholar] [Publisher Link]