Flautim: A Federated Learning Platform using K8S and Flower

Pedro H. S. S. Barros
Marcelo Q. A. Oliveira
pedro.barros@dcc.ufmg.br
marcelo@ifal.edu.br
Federal University of Minas Gerais
Belo Horizonte, Brazil

Omid Orang
Felipe A.R. da Silva
omid.orang@dcc.ufmg.br
fars@ufmg.br
Federal University of Minas Gerais
Belo Horizonte, Brazil

Fabricio J. Erazo-Costa
Allana Tavares Bastos
fabricioerazo@ufop.edu.br
allana-tb@ufmg.br
Federal University of Minas Gerais
Belo Horizonte, Brazil

Petrônio C. L. Silva
Glauber Soares dos Santos
petronio.candido@ifnmg.edu.br
glaubersoares@ufmg.br
Federal Institute of Northern Minas Gerais,Januária Campus, Brazil
Federal University of Minas Gerais
Belo Horizonte, Brazil

Antonio A. F. Loureiro
Martín Gómez Ravetti
Marcelo Azevedo Costa
loureiro@dcc.ufmg.br
martin@dcc.ufmg.br
macosta@ufmg.br
Federal University of Minas Gerais
Belo Horizonte, Brazil

Frederico Gadelha Guimarães
Heitor S. Ramos
fredericoguimaraes@ufmg.br
ramosh@dcc.ufmg.br
Federal University of Minas Gerais
Belo Horizonte, Brazil

ABSTRACT

Federated learning (FL) is a cutting-edge technology in artificial intelligence that preserves data privacy and security while reducing the cost of computation and communication. It transforms traditional centralized machine learning and deep learning approaches to enable decentralized model training without the need for data exchange. This work presents Flautim, the first implementation of an FL platform in Brazil and all around Latin America based on Kubernetes (K8S) and Flower framework. Flautim is designed for academic use, enabling researchers without a technical background to conduct FL experiments on this platform easily. Also, this platform allows for the development of applications involving data gathered from connected vehicles. Thus, this study aims to introduce this new FL platform, providing comprehensive details of its architecture.

KEYWORDS

Artificial intelligence, machine learning, deep learning, federated learning, data privacy and security

1 INTRODUCTION

Distributed learning [1] has recently emerged to overcome the challenges associated with centralized machine learning (ML) and deep learning (DL) techniques, including 1) limited access to advanced computing resources, 2) lack of expertise in managing and deploying tasks on high-performance computing systems, and 3) difficulties in centralizing data from diverse sources [2].

Federated Learning (FL) [3], as a particular case of distributed learning, is highly regarded for its ability to address privacy and security concerns through decentralized training. It involves a central server and multiple clients with their data. Each client trains a model locally and sends its update to the server. Then, the server aggregates updates to the global model and redistributes them. This iterative process facilitates collaborative learning without the necessity for data exchange, thereby maintaining data security and privacy.

A comprehensive review of FL in [4] highlights challenges, aggregation methods, and development tools. It emphasizes various tools and frameworks developed to simplify FL in real-world applications, including AI technology Enabler (FATE) [4], Substra [5], PyTorch [6], PySyft [7], PyGrid [4], Flower [8], OpenFL [9], TensorFlow federated (TFF) [10], IBM FL [11], NVIDIA Clara [4], among others.

Flower's functionality, efficiency, and user-friendliness make it the ideal choice for a unified FL approach. Flower is framework-agnostic, easily extensible, and allows a seamless transition from centralized to FL with minimal code changes [2]. Also, it is a flexible FL research framework [8] that excels in conducting large-scale experiments and accommodating a variety of heterogeneous device scenarios. Thus, the authors in [2] introduced AI4EOSC using Flower. AI4EOSC is an FL platform designed to advance AI, ML, and DL services and tools within the European Open Science Cloud (EOSC) using Consul, Nomad, and Traefic for secure, efficient orchestration and resource management. It allows users to deploy applications seamlessly through an automated system, with FL support using the Flower library. This setup effectively addresses challenges related to client connections and secure authentication, demonstrated through a practical implementation in a medical imaging scenario.

This paper presents Flautim, an FL platform built on Kubernetes (k8s) and Flower, designed for advanced AI, ML, and DL applications. Flautim marks the first initiative of its kind in Brazil and Latin America. Compared to other FL platforms like federated machine learning (FedML) [12], Flautim emphasizes user-friendliness and accessibility. While FedML primarily targets researchers and developers familiar with coding and technical configurations, Flautim's design is inspired by centralized approaches like PyTorch, which helps implement FL transparently and straightforwardly. Despite

In: XXII Workshop de Ferramentas e Aplicações (WFA 2024). Anais Estendidos do XXX Simpósio Brasileiro de Sistemas Multimídia e Web (WFA'2024). Juiz de Fora/MG, Brazil. Porto Alegre: Brazilian Computer Society, 2024.

ISSN 2596-1683

requiring some technical abilities, Flautim significantly simplifies the process using a familiar coding environment. This approach ensures that data remains on local devices, compliant with global data protection regulations like General Data Protection Regulation [1] (GDPR) and Brazilian General Data Protection Law [2] (LGPD).

Flautim aims to mitigate the gap between the technical complexity of FL and the practical needs of enterprises and researchers. By promoting a responsible, sustainable, and ethical approach to AI development, Flautim provides essential computational resources and a secure collaborative environment, representing an advancement in the field of FL and improving issues of privacy, collaboration, and intelligent decision-making in the digital age. The full description of Flautim's structure is explained in Section 3.

Thus, the rest of this paper is organized as follows: Section 2 explores the significance of the FL platform, Section 3 outlines the architecture of Flautim, Section 4 provides a guideline for running applications on this FL platform, and the last section concludes the paper and suggests the future directions.

2 WHY IS AN FL PLATFORM NEEDED?

FL has emerged as a decentralized and distributed ML technology, differing from conventional ML training by eliminating the need for direct data exchange between clients and the server. Instead, clients perform local training and only share model parameters or errors with the server, aggregating them to create a global model sent back to the clients. This approach preserves privacy and data security by keeping client data locally. Additionally, in applications with numerous sensors that generate vast amounts of data (Big Data), FL enhances scalability, efficiency, and communication latency. Furthermore, a collaborative methodology can be improved with edge devices, ensuring the training process happens in the clients.

Figure 1 shows the interface of Flautim. It features a search bar for finding institutions by name or CNPJ, with an option to show only active institutions. The main section displays a set of fantasy brand institutions with logos and identification numbers. This interface enables managing and interacting with various institutions and their FL models.

Figure 2 shows the detailed view of an institution within the Flautim platform. The institution, represented by a logo and name, has tabs for "Projects", "Users", and "Institution Data". The "Projects" tab is active, displaying a project with a placeholder name, modification date, brief description, and metrics for models, datasets, and experiments. This screen enables users to manage and view detailed information about the institution's projects, users, and data.

Implementing FL experiments in enterprise and educational environments is complex. It requires a robust infrastructure with high-performance computing and expert knowledge in various areas, including developing architecture, frameworks, communication protocols, and privacy-preserving techniques. This requires the involvement of professionals from multiple fields, such as data science, computer science, IT architecture, compliance, and legal.

[1] https://gdpr-info.eu/

[2] https://iapp.org/resources/article/brazilian-data-protection-law-lgpd-english-translation/

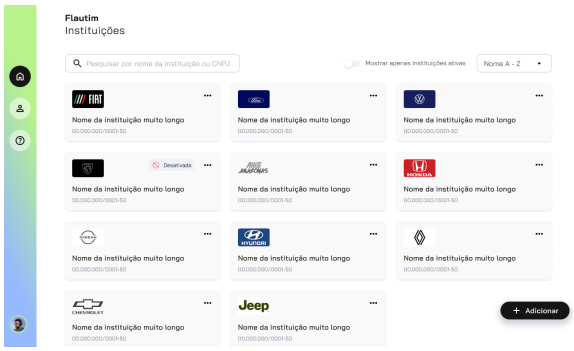

Figure 1: Figure shows the main interface of the Flautim platform, displaying a set of various fantasy brand institutions. Each card includes the institution's logo, name, and identification number.

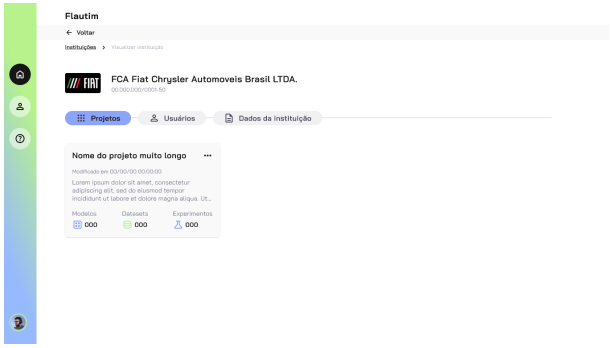

Figure 2: This figure presents the detailed view of an institution in Flautim, highlighting tabs for "Projects", "Users", and "Institution Data". Flautim shows a project with placeholder text for its name, description, last modified date, and the number for models, datasets, and experiments.

Nowadays, FL service providers are predominantly based in North America and Europe. Therefore, an FL platform entirely developed in Brazil is crucial to ensure that Latin America remains at the forefront of digital innovation. This would enable the region to actively participate in advancing cutting-edge technological solutions. Consequently, the Flautim FL platform provides services for the government, enterprise, healthcare, financial, and research sectors.

3 FLAUTIM PLATFORM ARCHITECTURE

Designing an FL framework in production requires a physically and logically distributed infrastructure. While several programming frameworks facilitate the development of federated models, the supporting infrastructure still needs to be planned and installed. Depending on the federated approach, this task can be challenging.

The Flower[3] framework is a Python programming API that offers a user-friendly interface for FL across major DL frameworks, such as PyTorch. It facilitates access to infrastructure services, including

[3] https://flower.ai/

exchanging the local models to the server and reception of averaged models, as well as productivity features like benchmarking, instrumentation, etc. The programming API provides the interface for the FL service and may be available in various programming languages and technologies, expanding its user base. In addition, the logical components of the infrastructure manage the physical cluster, distribute tasks, and monitor activities, often blurring the lines between the logical and physical layers due to their intrinsic integration.

The physical layer can be divided into the client and the server. The client-side includes devices or computers that run local models of federated applications. The server side, in contrast, follows the architecture of high-performance and high-availability clusters, which are common in cloud computing infrastructures like those used in Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS) products such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, but on a smaller scale.

After implementing the internal layer of the logical architecture on the physical architecture, it is crucial to ensure that the platform meets its non-functional requirements, such as performance, availability, and robustness. Managing the various software components and their requirements across multiple processing nodes in a cluster is challenging without employing modern infrastructure engineering techniques, particularly the containerization paradigm.

Containerization [13, 14] represents a paradigm shift in application development, deployment, and maintenance, replacing the traditional virtualization paradigm. Unlike traditional virtualization, which requires emulating an entire computational stack, containerization only emulates the operating system stack. This approach conserves memory and processing power, allowing more containers to run simultaneously on the same machine. Thus, containerization can be seen as a streamlined and optimized form of virtualization. A container platform manages the execution of containers, providing isolation, performance, and security. Docker [4] is the most popular container platform, significantly contributing to making containerization a standard in software infrastructure.

Once containerization reaches a large scale, managing the multitude of simultaneous containers running across multiple cluster nodes becomes challenging. Additionally, manual intervention is required to reboot or redeploy when a container crashes. This complexity necessitates the container orchestration tools to automate these tasks and ensure efficient management.

The Kubernetes[5] system, also known as k8s, is a management tool designed for large-scale containerized application environments running in clusters. K8s is utilized in various contexts, including High-Performance Computing (HPC) clusters [15], Cloud Computing [16], microservices [17], among others. K8s can manage Docker containers and other technologies compliant with the Container Runtime Interface (CRI)[6] protocol. K8s offers both web APIs and command-line interfaces (CLI) to facilitate the creation and deployment of containers. Central to its architecture is the principle of "Infrastructure as Code" (IaC) [18]. This approach involves storing all cluster configurations in plain text files using the YAML format, enhancing readability and simplifying deployment. As a result, the web API and CLI are predominantly employed for maintenance and monitoring tasks.

The components of the FL platform should be logically decomposed and implemented using K8s objects such as services and jobs. Each object represents one or more containers, known as Pods, with specific properties. These Pods are deployed with multiple replicas across the nodes of the cluster, ensuring redundancy, which is crucial for load balancing and high availability.

The FL Platform solution must manage load balancing, self-healing, and monitoring of the containers for all logical components across the cluster's physical nodes. Handling these essential maintenance tasks alleviates the burden on the operations team, allowing them to focus on more strategic aspects of platform development.

The proposed platform empowers automotive manufacturers and the software industry to develop federated applications for connected cars that evolve transparently based on user experiences. It is backed by a robust infrastructure to meet high processing and network demands. In conclusion, developing, deploying, and maintaining a scalable, centralized FL platform requires the capability to manage distributed software running in extensive computational clusters. Effective planning for this platform necessitates a comprehensive understanding of the logical and physical components and their interactions. Managing this complex environment involves utilizing modern technologies such as the containerization paradigm for software deployment and engineering tools like K8s, which ensure the infrastructure's maintenance, monitoring, and self-healing.

4 RUNNING AN APPLICATION IN THE PLATFORM

As the platform aims to facilitate experimentation of FL experiments, running applications is straightforward to anyone familiar with training machine learning models. The platform's interface guides users to input three essential files: a CSV dataset with an associated Dataset class, a Python script (.py) containing an API call for the model, and another Python script (.py) specifying the model's hyperparameters. Figure 3 shows the interface for starting an experiment in the Flautim platform. In the "Experimentos" section, users can input the experiment's name, add optional details, and select a model and dataset from dropdown menus previously added to Flautim. They can also upload API code and hyperparameter configuration files.

Upon clicking "Run Experiment", users can monitor the training progress, estimated completion time, and detailed logs on the "Log" page. After training, logs remain accessible, and the "Outputs" page displays plots of the model's accuracy. The trained model can be downloaded from the "Models" page.

Figure 4 shows the results interface for an experiment in the Flautim platform. This Figure indicates the user is viewing the output data of "Experimento01." The interface includes tabs for "Visão Geral", "Saídas", "Modelos", and "Log", with the "Outputs" tab active. It displays the training status, showing a duration of 00:02:35 and 100% progress (5/5 rounds completed). The performance metrics are provided, including a graph showing the loss over time, and users can download the metric's file. This screen helps users monitor and analyze experiment results.

[4] https://www.docker.com/
[5] https://kubernetes.io/
[6] https://github.com/kubernetes/cri-api

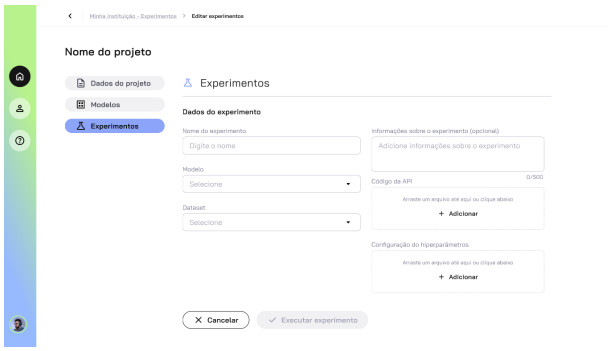

Figure 3: This figure illustrates the interface for starting an experiment in Flautim.

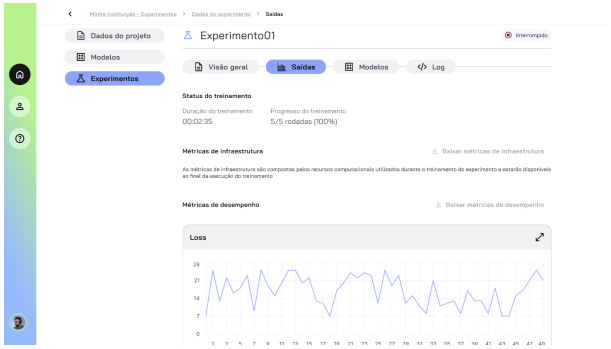

Figure 4: This figure shows the results interface for an experiment in our FL platform. It includes training status details, such as duration and progress, along with infrastructure and performance metrics sections. A graph displays the loss over time, and options to download metrics are available.

Finally, all figures were created using Figma [7]. The final version of our platform may vary slightly from the images provided.

5 CONCLUSION

This paper presents Flautim, the pioneering FL platform in Brazil and Latin America, built on K8s and the Flower framework. Flautim is designed to be accessible, allowing researchers and automotive manufacturers to conduct FL experiments and develop applications easily, even without extensive technical expertise. Released under the BSD license, Flautim can potentially support both commercial and open-source use, maintaining data locally and complying with global data protection regulations. This platform simplifies the deployment of FL systems while ensuring data privacy and security, representing a significant step forward in making FL more practical and widespread, promoting ethical AI development, and facilitating collaborative research efforts. For future work, we intend to conduct a broad range of experiments to validate the applicability of this platform across various federated applications.

[7] https://www.figma.com/pt-br/

ACKNOWLEDGMENT

This work was partially funded by São Paulo Research Foundation (grant #2023/00721-1), Conselho Nacional de Desenvolvimento Científico e Tecnológico (grant #312682/2021-2) and Fundação de Amparo a Pesquisa do Estado de Minas Gerais (grant #APQ-00426-22).

REFERENCES

[1] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," vol. 53, no. 2, 2020.
[2] J. Sáinz-Pardo Díaz, A. Heredia Canales, I. Heredia Cachá, V. Tran, G. Nguyen, K. Alibabaei, M. Obregón Ruiz, S. Rebolledo Ruiz, and A. López García, "Making federated learning accessible to scientists: The ai4eosc approach," ser. IH&MMSec '24. New York, NY, USA: Association for Computing Machinery, 2024.
[3] J. Konecný, H. McMahan, F. Yu, P. Richtárik, A. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016.
[4] B. S. Guendouzi, S. Ouchani, H. E. Assaad, and M. E. Zaher, "A systematic review of federated learning: Challenges, aggregation methods, and development tools," *Journal of Network and Computer Applications*, p. 103714, 2023.
[5] M. Galtier and C. Marini, "Substra: a framework for privacy-preserving, traceable and collaborative machine learning," 10 2019.
[6] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.
[7] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose, T. Ryffel, Z. N. Reza, and G. Kaissis, *PySyft: A Library for Easy Federated Learning*, 06 2021, pp. 111–139.
[8] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.
[9] G. Reina, A. Gruzdev, P. Foley, O. Perepelkina, M. Sharma, I. Davidyuk, I. Trushkin, M. Radionov, A. Mokrov, D. Agapov, B. Edwards, M. Sheller, S. Pati, P. Moorthy, S.-h. Wang, P. Shah, and S. Bakas, "Openfl: An open-source framework for federated learning," 05 2021.
[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.
[11] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, M. Sinn, M. Purcell, A. Rawat, T. Minh, N. Holohan, S. Chakraborty, S. Whitherspoon, D. Steuer, L. Wynter, H. Hassan, and A. Abay, "Ibm federated learning: an enterprise framework white paper v0.1," 07 2020.
[12] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu *et al.*, "Fedml: A research library and benchmark for federated machine learning," *arXiv preprint arXiv:2007.13518*, 2020.
[13] J. Watada, A. Roy, R. Kadikar, H. Pham, and B. Xu, "Emerging trends, techniques and open issues of containerization: A review," *IEEE Access*, vol. 7, pp. 152 443–152 472, 2019.
[14] O. Bentaleb, A. S. Belloum, A. Sebaa, and A. El-Maouhab, "Containerization technologies: Taxonomies, applications and challenges," *The Journal of Supercomputing*, vol. 78, no. 1, pp. 1144–1181, 2022.
[15] A. M. Beltre, P. Saha, M. Govindaraju, A. Younge, and R. E. Grant, "Enabling hpc workloads on cloud infrastructure using kubernetes container orchestration mechanisms," in *2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC)*. IEEE, 2019, pp. 11–20.
[16] S. Hardikar, P. Ahirwar, and S. Rajan, "Containerization: cloud computing based inspiration technology for adoption through docker and kubernetes," in *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE, 2021, pp. 1996–2003.
[17] L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying microservice based applications with kubernetes: Experiments and lessons learned," in *2018 IEEE 11th international conference on cloud computing (CLOUD)*. IEEE, 2018, pp. 970–973.
[18] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, "A systematic mapping study of infrastructure as code research," *Information and Software Technology*, vol. 108, pp. 65–77, 2019.