Federated Outlier Detection for Astronomical Data: Performance Analysis on Commercial Clouds*

Camila Lopes[1], Wesley Ferreira[1], Julia Gschwend[2], Luiz Nicolaci da Costa[2] Rafael Ferreira da Silva[3], Marta Mattoso[4], Aline Paes[1], Daniel de Oliveira[1]

[1]Institute of Computing – Universidade Federal Fluminense (IC/UFF)

[2]Laboratório Interinstitucional de e-Astronomia (LIneA)

[3]National Center for Computational Sciences, Oak Ridge National Lab

[4]Universidade Federal do Rio de Janeiro (COPPE/UFRJ)

{camila_ol,wesleyferreira}@id.uff.br, {alinepaes,danielcmo}@ic.uff.br

{julia,ldacosta}@linea.org.br, silvarf@ornl.gov, marta@cos.ufrj.br

Abstract. *Astronomical surveys such as the Dark Energy Survey (DES) and the upcoming Legacy Survey of Space and Time (LSST) produce massive volumes of observational data, demanding scalable and efficient data analysis techniques. Among these, Machine Learning (ML) has become a key tool for extracting patterns and detecting anomalies in large astronomical catalogs. However, traditional centralized ML approaches are impractical in this scenario, due to data transfer bottlenecks, storage constraints, and privacy concerns. Federated Learning (FL) offers a decentralized alternative by training models across distributed data sources, reducing transfer costs and preserving data locality. However, configuring and deploying FL workflows is challenging due to client heterogeneity and data distribution. This paper explores the use of FL for outlier detection in large astronomical catalogs, using DES as a proxy for LSST. We emulate FL deployments on Amazon Web Services (AWS) cloud, evaluating various configurations of compute resources. Our results evidence the trade-offs between training time and financial cost, providing insights into the configuration of FL workflows for large-scale LSST data.*

1. Introduction

Over the past decades, astronomical surveys, *e.g.*, Dark Energy Survey (DES) [Abbott et al. 2018], have transformed our understanding of the universe by collecting data on millions of objects across regions of the sky [Mickaelian 2016]. These surveys produce large catalogs with hundreds of attributes, *e.g.*, positions, brightness, spectral properties, and other physical characteristics, associated with a plethora of objects like stars and galaxies [Jurić et al. 2015]. While astronomers can still rely on visual inspection and expert analysis to draw conclusions from these catalogs, the data volume has made such methods impractical.

To help address these challenges, Machine Learning (ML) techniques have been used in astronomy for many years in several tasks, *e.g.*, the classification of astronomical transients [D'Isanto et al. 2016], light curve analysis and classification [Mahabal et al. 2017], and asteroid identification [Alves et al. 2025]. Beyond these well-established applications, ML methods are also used as tools for exploratory analysis, such as the detection of outliers in the

*This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. The authors would also like to thank CNPq and FAPERJ.

catalogs [Covey et al. 2007]. Indeed, one of the potential uses of astronomical catalogs is the identification of rare or unusual objects that may hold the key to scientific discoveries.

Although the application of ML techniques in astronomy represents a step forward, the rapid increase in the scale of astronomical catalogs and the volume of generated data introduces new challenges. The complexity of this scenario has grown with the start of operations at the Vera C. Rubin Observatory's Legacy Survey of Space and Time (LSST) [Ivezić et al. 2019]. Equipped with a 3.2-gigapixel camera, the observatory can capture images at an unprecedented scale. Each night, it is expected to generate approximately 20 terabytes of raw data. This volume of data is transferred to a US Data Facility at SLAC[1], where it undergoes processing to produce large-scale astronomical catalogs.

The volume of data produced by LSST not only makes traditional visual inspection methods impractical but also imposes limitations on the training of traditional ML models [Savić et al. 2023]. Model training typically requires that the entire dataset be accessible in one place, which becomes challenging as data volumes grow. LSST catalogs are expected to span hundreds of terabytes, creating critical storage demands and data transfer bottlenecks when attempting to centralize the data for ML training purposes. Although advanced network infrastructure exists to ease the transfer of data from the Rubin Observatory to SLAC and partner institutions, only a few Independent Data Access Centers (IDACs)[2] have the technical capacity and the authorization to store and process such catalogs.

To address these challenges, Federated Learning (FL) [McMahan et al. 2017] has emerged as a promising paradigm for distributed ML. FL enables the decentralized training of models by allowing data to remain on local clients (in the context of this paper, the IDACs would serve as these clients), while only model updates, *e.g.*, weights of a neural network or centroids in clustering algorithms, are shared with a server for aggregation. This approach minimizes the need for large-scale data transfers and preserves data locality. FL has shown its potential in various fields, *e.g.*, healthcare, where privacy and data distribution are critical concerns. Still, its application to large-scale scientific data analysis, especially in the field of Astronomy, remains unexplored and represents an open field [Razmi et al. 2024].

However, training ML models for astronomical applications using FL is far from straightforward. Several factors can affect both training performance and the quality of the trained models. For example, if a subset of clients presents limited computational resources or restricted bandwidth, they may delay the transmission of model updates, thus increasing the training time. Similarly, if the central server has insufficient computational capacity, the aggregation of updates and synchronization with clients can become a bottleneck. Moreover, the partitioning of astronomical catalogs among clients must be handled carefully to ensure that the data distribution is as balanced as possible, a requirement that may be difficult to achieve in practice. Poorly balanced or skewed data distributions can hinder the convergence of the global model and increase the risk of model degradation. These challenges highlight the complexity of applying FL effectively to large-scale astronomical catalogs.

In this paper, we evaluate the feasibility of using FL for the problem of object outlier detection in the large-scale DES Data Release 2 (DR2) [Abbott et al. 2021]. We use the `FLARE+Prov`, an extension of `NVIDIA FLARE` [Roth et al. 2023], to deploy and evaluate the FL performance with a particular focus on preparing for the data volume expected from

[1]`https://www6.slac.stanford.edu/lsst`
[2]`https://www.linea.org.br/idac`

the LSST. We use the DES dataset as a proxy for LSST's data, given that the LSST's IDACs are planned to begin operations only in early 2026. Our study seeks to proactively evaluate FL by emulating the distributed environment and the heterogeneous data expected. Specifically, we emulate deployments using Amazon AWS as a platform. The core idea of our work is to explore various deployment scenarios that differ in terms of client and server performance characteristics and data distribution strategies. Through this exploration, we aim to analyze the total training time required by FL-based workflows, as well as the financial costs associated with these deployments. By doing so, we plan to generate insights into the practical trade-offs and constraints involved in executing federated outlier detection tasks at the scale anticipated for LSST.

2. Outlier Detection in Astronomical Surveys

Outlier detection task [Covey et al. 2007] plays a critical role in astronomy, particularly in large-scale surveys where catalogs contain millions (or even billions) of objects with different characteristics (Figure 1 - left). Identifying "anomalous" or rare objects can provide insights or highlight areas of the sky that require further visual analysis. Astronomical outlier detection is typically based on observational features such as magnitudes and colors. *Magnitudes* in astronomical catalogs refer to the brightness of an object measured using specific filters that consider parts of the electromagnetic spectrum [Fukugita et al. 1996]. Common used filters are optical ones, *e.g.*, u, g, r, i, z, and infrared, *e.g.*, Y, J, H, K. In the structured catalog, each astronomical object typically has a measured magnitude for each filter, commonly named as `g_mag`, `r_mag`, `i_mag`, `z_mag`, *etc.* Specifically in the experiments considered in this paper we used the Auto Magnitude (mag_auto) [Bertin and Arnouts 1996], which is based on an elliptical aperture that adapts to the shape and size of each object individually using the Kron radius [Kron 1980] to contain most of the object's flux while minimizing contamination from near objects.

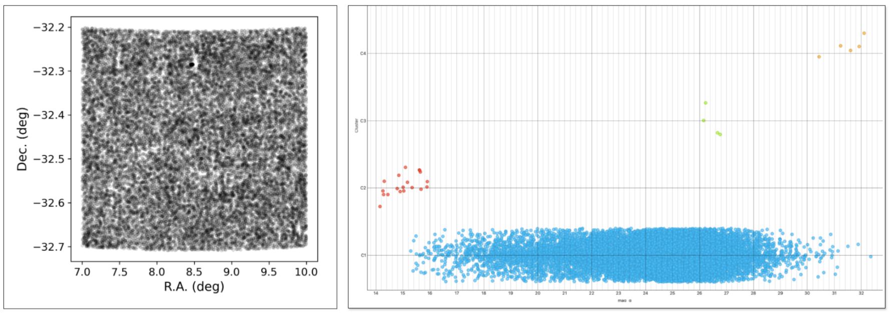

Figure 1. Spatial distribution of objects (left) and clusters (right) restricted to a small sky region with coordinates $(7., -32.2, 10, -32.2, 10, -32.7, 7., -32.7)$**.**

Colors, in turn, are derived from the differences in magnitudes between photometric bands (*e.g.*, $g-r$, $r-i$, $i-z$) and serve as representations for stellar temperature, age, and composition. Objects with extreme color combinations may represent high-redshift galaxies or outliers. Similarly, *magnitude* helps isolate objects that are either much fainter or brighter than expected for a given population, which can indicate the presence of flares, supernovae, or saturated observations. In order to detect outliers, ML methods such as K-Means [MacQueen 1967] can be applied using the aforementioned features. These algorithms identify regions of low-density occupancy or sparse neighborhoods, which may be indicative of outlier objects. K-Means clustering, particularly, can be leveraged for unsupervised outlier detection by identifying data points that lie far from their assigned cluster centroids, based on the assumption that normal instances form dense groups while anomalies appear as isolated points. This approach is computationally efficient and well-suited for large-scale datasets where labeled anomalies are scarce. For instance, in the feature space defined by $(g - r, r - i, i - z, r\_mag, g\_mag, i\_mag, z\_mag)$, clusters generated from the DES DR2 dataset, restricted to a small sky region with coordinates $(7., -32.2, 10, -32.2, 10, -32.7, 7., -32.7)$, are presented in Figure 1 (right). In the upcoming LSST survey, the massive data volume will make centralized outlier detection computationally prohibitive. Therefore, scalable and distributed approaches such as FL should be explored to enable in-situ anomaly detection while preserving data locality.

3. A Brief Tour on Federated Learning

Federated Learning (FL) [McMahan et al. 2017] is a decentralized ML paradigm that enables model training across multiple clients (that can be computers, VMs, clusters, mobile phones) without the need to access the complete dataset. This strategy has gained attention, particularly since it avoids the transmission of raw data between participants, avoiding unnecessary data transfers and maintaining privacy. In FL, only model updates are shared with a central server [Nair et al. 2022].

FL follows a well-defined and iterative workflow. The training is organized in multiple rounds. At each round, the server defines the type of model to be trained (*e.g.*, a CNN) and distributes it to the clients with the associated hyperparameters. Each client performs local training using local data, without sharing the raw data with other clients or with the server. When the local model is trained, clients send model updates, *i.e.*, metadata associated with the trained model, such as neural network weights, back to the server. The server aggregates these updates to generate a new global model using an aggregation strategy. A widely adopted method is *Federated Averaging (FedAvg)*, in which each client performs multiple steps of local stochastic gradient descent on its data, and the server computes a weighted average of the resulting model parameters to update the global model. This updated model is then broadcast back to the clients to initiate the next communication round.

Besides the hyperparameters of the ML method (*e.g.*, dropout rate, loss), the FL workflow also has specific parameters, *e.g.*, the number of rounds T, the total number of clients K, the fraction C of clients selected to participate in each round, and the batch size B, which defines the number of training examples processed per round. FL can be categorized into two classes: *Cross-Device* and *Cross-Silo* [Yang et al. 2019]. In the Cross-Device, clients are typically mobile devices, and the system may scale to millions of participants. In contrast, Cross-Silo, which is the focus of this paper, involves a smaller number of stable clients, *i.e.*, no churn is expected. Several frameworks support the execution of FL workflows, including NVIDIA FLARE [Roth et al. 2023] and Flower [Beutel et al. 2020]. These frameworks offer integration with popular ML libraries such as Keras, TensorFlow, and PyTorch, enabling seamless deployment across multiple environments, including cloud, mobile, on-premise, and edge. In this paper, an extended version of `NVIDIA FLARE` is used as a framework.

4. Methodology

This section describes the materials and methods employed to evaluate a series of deployment scenarios, each with specific client–server configurations and data distribution strategies. These experiments aim to quantify the total training of FL workflows and to uncover trade-offs and operational constraints inherent to executing federated outlier detection tasks at the scale projected for the LSST.

4.1. FL Framework

All experiments were conducted using an extended version of the `NVIDIA FLARE` framework [Roth et al. 2023] named `FLARE+Prov`. `FLARE+Prov` extends the architecture of the original `FLARE` by adding new components to capture provenance data and automatically deploy the server and clients (Figure 2). The FL workflow in `FLARE` is based on interactions between FL Clients and a FL Server. The FL Server coordinates the execution of the workflow, *i.e.*, it receives local model updates from clients, performs aggregation, and distributes the updated global model back to clients for another training round. On the other hand, the FL Clients train the model *in situ*, consuming their local and private data, and send metadata regarding the trained model back to the server. For local training, each client can invoke ML libraries such as PyTorch and TensorFlow.

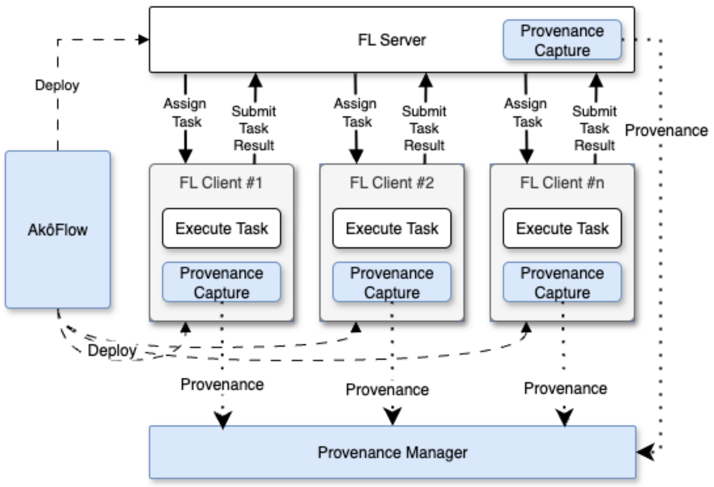

Figure 2. The architecture of the `FLARE+Prov` framework.

`FLARE` was extended into two directions (blue components in Figure 2): (i) provenance management [Herschel et al. 2017] and (ii) automatic deployment. Regarding provenance management, the architecture was extended by adding provenance capture components on both clients and the server. The idea follows a similar approach adopted in prior work [Lopes et al. 2023], where hyperparameters used during each training round, intermediate and final performance metrics, and system-level data (*e.g.*, runtime and resource usage) are captured and stored in a structured database. This database stores the complete data derivation path during distributed training, thereby ensuring end-to-end traceability of model development and reproducibility. Regarding the automatic deployment, `AkôFlow` [Ferreira et al. 2024] was integrated into the architecture to create a virtual environment where the FL workflow executes. `AkôFlow` instantiates multiple virtual machines (VMs) in the cloud environment, deploys clients and servers within those VMs, executes the FL workflow, and monitors resource consumption, such as CPU. The source code will be made available at `https://github.com/UFFeScience/flare_prov`.

4.2. Data

The experimental data were obtained from the Dark Energy Survey (DES) Data Release 2 (DR2), an astronomical catalog containing photometric observations for millions of celestial objects. In this paper, a subset of the catalog was divided into ten fragments, each representing a different 2×2 degree region of the sky. Each fragment has approximately 450,000 objects and is stored in a separate CSV file. These files include positional and photometric attributes for each object. The attribute `coadd_object_id` serves as an identifier for catalog entries. The attribute `ra` denotes Right Ascension (RA), specifying the east-west position on the celestial sphere, while `dec` indicates Declination (Dec), measuring the angular distance north or south of the celestial equator.

In the used dataset, photometric measurements are represented in multiple bands, each corrected for the effects of interstellar dust extinction (*i.e.*, dereddened). Specifically, `mag_auto_g_dered` denotes the total magnitude in the g-band (green optical); `mag_auto_r_dered` refers to the total magnitude in the r-band (red optical); `mag_auto_i_dered` corresponds to the near-infrared magnitude in the i-band; `mag_auto_z_dered` provides the total magnitude in the z-band (far red/near-infrared); and `mag_auto_y_dered` indicates the magnitude in the y-band, which is the reddest band in the optical/near-infrared range. To enrich the input features for the outlier detection process, we computed a set of colors based on the differences between magnitudes in adjacent photometric bands. Specifically, we derived the following color features: $g - r$, $r - i$, and $i - z$. These colors are essential for distinguishing variations in the spectral energy distributions of objects, thereby enhancing the sensitivity of the ML-based outlier detection algorithms.

4.3. Experimental Setup

To investigate the feasibility of applying FL for the problem of astronomical outlier detection, we designed three types of experiments focusing on different aspects of the FL workflow: (i) server performance under varying configurations, (ii) the effects of client heterogeneity, and (iii) the impact of data volume and distribution among clients. Each analysis is described in detail in the following. Federated outlier detection was conducted in all experiments by executing the K-Means clustering algorithm [MacQueen 1967] in each client. Following local model training, each client sends its computed centroids and the count of local data samples used for training to a central server. The server then performs aggregation on the centroids, using a weighted average based on sample counts, analogous to the FedAvg method.

The first experiment evaluates the performance of various server configurations. The FL server is deployed on three types of VMs: (i) m7i.xlarge, a general purpose VM with 4 vCPUs, 16 GiB RAM, and a financial cost of US$0.22176 per hour; (ii) c7i.xlarge, a compute optimized VM with 4 vCPUs, 8 GiB RAM, and a financial cost of US$0.19635 per hour; and (iii) r7i.xlarge, a memory optimized VM with 4 vCPUs, 32 GiB RAM, and a financial cost of US$0.29106 per hour. In this experiment, all clients use homogeneous hardware (m7i.xlarge VMs). Each client receives one fragment of the DES catalog described in Subsection 4.2, resulting in a total of 10 clients. This controlled setup isolates and evaluates the server's impact on FL workflow performance.

The second experiment evaluates the impact of client heterogeneity on the FL training workflow. The server's VM type is fixed, while the clients' computational resources vary. Both homogeneous and heterogeneous client pools are defined to analyze the performance implications of these configurations. Each client receives a different fragment of the dataset, consistent with the previous experiment. This experiment evaluates whether variations in client resources, such as the number of vCPUs or memory, influence training duration or convergence behavior. The configurations being evaluated include: (i) 10 m7i.xlarge clients,

(ii) 10 c7i.xlarge VMs, (iii) 10 r7i.xlarge VMs, (iv) 5 m7i.xlarge VMs and 5 c7i.xlarge VMs, (v) 5 m7i.xlarge VMs and 5 r7i.xlarge VMs, (vi) 5 c7i.xlarge VMs and 5 r7i.xlarge VMs, and (vii) 3 m7i.xlarge VMs, 3 c7i.xlarge VMs, and 4 r7i.xlarge VMs.

The third experiment evaluates how the distribution of data fragments among clients affects the FL workflow performance and outcomes. The experiment increases the size of each data fragment assigned to clients, thereby reducing the total number of clients, and investigates alternative strategies for merging data into larger fragments. The first strategy merges two spatially adjacent fragments into a single file for assignment to a client. The second strategy randomly merges dataset fragments, disregarding spatial distribution, before assignment. Training performance is evaluated under these allocation strategies to evaluate whether spatially adjacent fragments within clients enhance or hinder the federated outlier detection process. The implications for model convergence are also analyzed in both scenarios. The experimental environment consisted of one r7i.xlarge VM as the server and 5 r7i.xlarge VMs assigned as clients.

5. Results

This section presents and analyzes the results obtained from the three experiments previously described. Regarding the first experiment (performance analysis of various server configurations), Figure 3(a) presents the makespan (average of 5 executions), and Figure 3(b) depicts the financial costs. Analyzing the makespan shows that the choice of server VM type impacts the performance of the FL workflow. In particular, it can be observed that memory-optimized and memory-balanced VMs achieve superior performance compared to compute-optimized VMs, highlighting the role of RAM in determining execution efficiency.

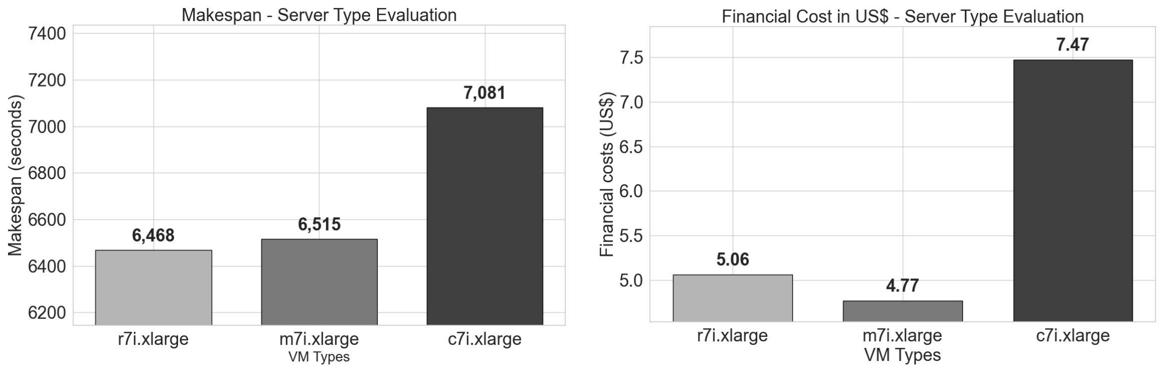

Figure 3. Server Evaluation (a) Makespan (b) Financial Costs

The reason behind this behavior is that the model evaluation step of the FL workflow affects both the makespan and financial costs. In each round, the FL workflow computes the silhouette score to evaluate model performance. This procedure requires determining the average distance from each data point to all other points within the same cluster and those in the nearest neighboring cluster. Each client performs this memory-intensive procedure, which is especially demanding for datasets with hundreds of thousands of data points, as analyzed in this study. VMs with greater memory capacity process these large-scale computations more efficiently, resulting in improved performance.

The results demonstrate that VMs with larger memory present better performance. Nevertheless, it is worth noting that the makespan achieved by the r7i.xlarge (memory-optimized) and m7i.xlarge (memory-balanced) VMs was relatively similar, despite differences in hardware configurations. From a cost-performance perspective, the m7i.The xlarge VM is the more advantageous option, as it offers nearly the same level of performance as the r7i.xlarge but at a lower financial cost. Although the differences in both makespan and cost may appear small in absolute terms, they become critical when considering the operational context of the FL workflow for astronomy. Specifically, within the scope of the LSST, new observational data will be generated every three days over ten years. Consequently, the workflow will need to be executed repeatedly, thereby amplifying the cumulative impact of even small performance and cost differences.

While server memory impacted overall performance, CPU usage was also measured during training. Figure 4 presents CPU usage for one representative configuration, selected due to space limitations. This pattern was consistent across all executions. Figure 4 shows CPU usage for the server and three clients: (i) client 1 shows typical usage, (ii) client 9 shows intermittent decreases, and (iii) client 10 maintained stable usage throughout training. None of the clients exceeded an average of 50% CPU usage, indicating that memory is the primary resource requirement for both clients and the server, rather than CPU.

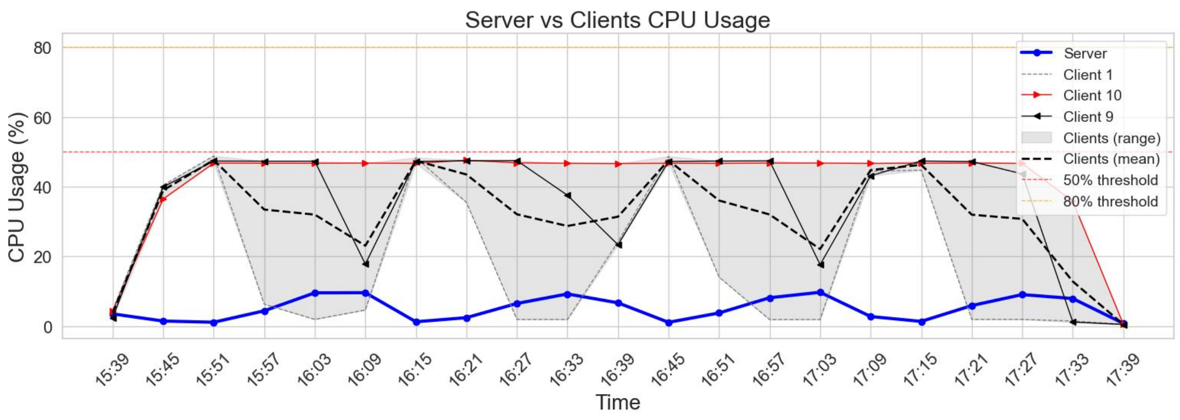

Figure 4. Server *vs.* Client CPU Usage

In the second experiment, which evaluates the impact of client heterogeneity on the FL workflow, Figure 5(a) shows the workflow makespan, and Figure 5(b) presents the associated financial costs. The results are consistent with those from the first experiment. Client pools composed exclusively of r7i.xlarge (memory-optimized) and m7i.xlarge (memory-balanced) VMs achieved lower makespan and financial cost. On the other hand, client pools that contain c7i.xlarge (compute-optimized) VMs resulted in increased makespan and cost. This result is attributed to the memory-intensive nature of the K-Means algorithm executed on each client, especially when processing large datasets. K-Means requires computation of the distance between n data points and each of the k centroids during each iteration. The implementation used in this study generates an $n \times k$ distance matrix, which can become substantial in size. For specific data fragments, approximately 4-5 GB of RAM is needed solely for storing the distance matrix. This memory constraint is particularly significant for the clients operating on the c7i.xlarge VM, which is limited to 8 GiB of RAM.

Regarding the third experiment, it addresses two primary research questions: (i) whether increasing the number of data points assigned to each client for outlier detection improves performance, and (ii) whether merging spatially adjacent fragments facilitates faster or more stable convergence of the ML model. Figure 6(a) presents the distribution of chunk

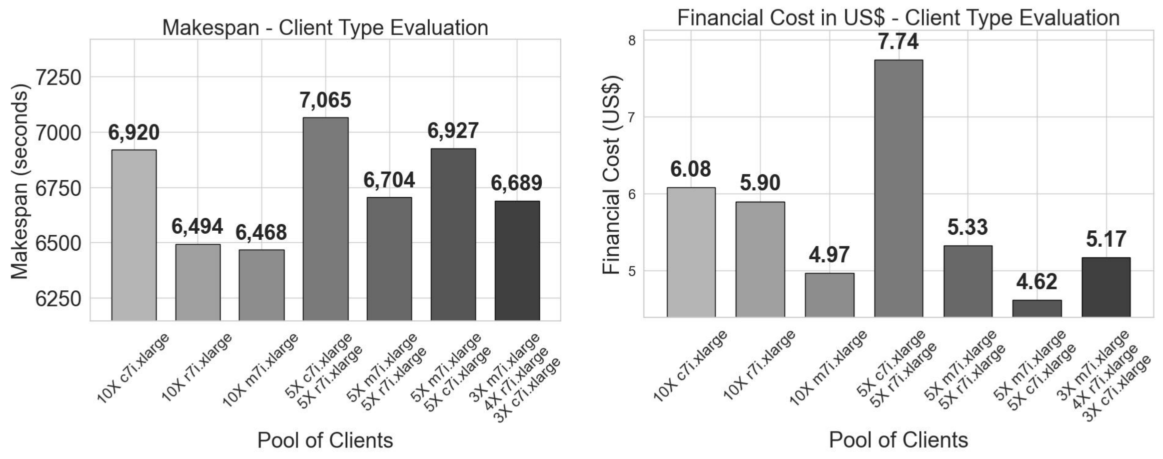

Figure 5. Client Evaluation (a) Makespan (b) Financial Costs

sizes processed by each client under three scenarios: the original 10 fragments left unmerged (Uniform), merging of two spatially adjacent fragments, and randomly merging of fragments.

One can observe that the chunk size in the case of spatially adjacent merging presents a high degree of variability. This happens because a very high density of objects is found in some areas of the sky, while others are relatively sparse. As a consequence, although spatial adjacency can benefit the K-Means (spatially adjacent data points are more likely to belong to the same cluster, thereby stabilizing centroid updates and reducing the number of required iterations), the resulting merged fragments can differ significantly in size. For example, in our experiments, some spatially adjacent fragments contained approximately 1.4 million objects (the largest fragment in Figure 6(a)). On the other hand, under the random partitioning strategy, the maximum fragment size was approximately 900,000 objects.

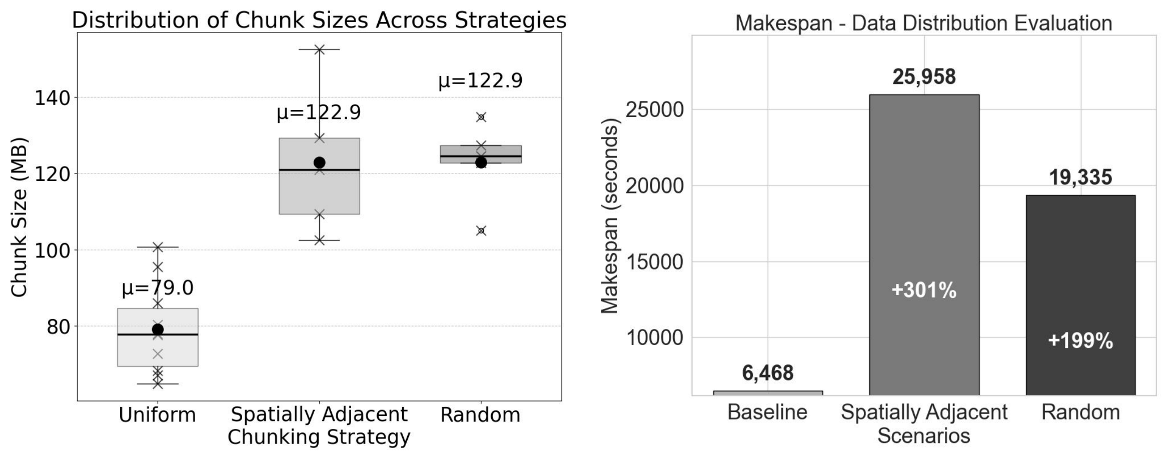

Figure 6. Data distribution evaluation (a) Chunk size (b) Makespan

This imbalance has important implications for FL. Specifically, clients assigned to very large fragments required more time to train their local models compared to clients with smaller fragments. These slower clients effectively became bottlenecks in the execution, impacting the makespan of the FL workflow. Although we expected that the makespan would increase when reducing the number of clients from 10 to 5, since each fragment would then contain roughly double the number of datapoints, Figure 6(b) shows a more nuanced outcome. In practice, the spatially adjacent strategy took 301% more time than the baseline configuration (10 clients with smaller fragments as in the first experiment). In comparison, the random partitioning strategy took 199% more time than the baseline.

These results provide two key insights. First, they demonstrate that even though spatial adjacency is beneficial for K-Means clustering, the size of the dataset, primarily driven by the density of points within each region, has a greater influence on performance. Second, they indicate that the makespan does not grow linearly with the number of data points processed per client. This implies that increasing fragment sizes must be approached with caution, carefully balancing the trade-off between spatial adjacency and dataset density. In other words, partitioning strategies must not only consider whether points are spatially adjacent, but also take into account the densities of objects across different regions to avoid performance degradation.

6. Related Work

Recent research on FL has focused on addressing challenges in decentralized ML, particularly in contexts that are similar to the one discussed in this paper. An issue in FL is data heterogeneity, which causes clients to store and process local datasets that differ in size, distribution, or statistical properties, thereby impacting the global model's convergence. Some approaches have been proposed to solve this problem. The Clustered Federated Learning (CFL) framework [Sattler et al. 2019] dynamically groups clients that present similar local data distributions, enabling the training of ML models that are fine-tuned to subsets of data rather than seeking convergence of a global model. FL has been adopted in a plethora of domains to perform tasks such as outlier identification. [Laridi et al. 2024] investigated federated anomaly detection using autoencoders with non-independent and identically distributed (non-IID) data, showing that decentralized learning can maintain robustness even under heterogeneous conditions. FL has been applied to domains such as finance and cybersecurity. [Aljunaid 2025] demonstrated its utility for detecting fraudulent financial transactions across distributed institutions, while [Cui et al. 2022] applied FL to enhance the security of Internet of Things (IoT) infrastructures against adversarial threats.

In Astronomy, FL has only recently begun to be explored, mainly as a strategy to overcome the high costs of data transfer. Astronomical surveys generate large-scale datasets, often reaching petabyte scale, which are infeasible to move between sites (a "site" may denote a single machine, a cluster, or even a data center). A NASA study demonstrated the use of FL for training models on distributed datasets located both on the International Space Station (ISS) and on Earth [Casaletto et al. 2022], thereby evaluating its applicability in space-based sensing environments. Other approaches have considered the broader systems perspective, such as the energy efficiency of FL in satellite constellations [Razmi et al. 2024], or the challenges posed by class imbalance and heterogeneous distributions in aerial and space networks [Dong et al. 2024]. Building upon these efforts, this paper extends the discussion of FL into a novel application: object outlier detection in large-scale astronomical catalogs. While anomaly detection has been studied in other domains, its application to astronomy in a federated context remains unexplored.

7. Conclusions

In this paper, we investigate the challenges of using FL for large-scale astronomical catalogs, with a particular focus on outlier detection using the DES DR2 dataset in preparation for the forthcoming LSST data volume. We evaluated how different deployment strategies, i.e., varying server configurations, client heterogeneity, and data distribution strategies, affect both makespan and cost when executing FL workflows in a cloud environment. Our study used the `FLARE+Prov` framework as an FL framework, which extends NVIDIA FLARE with provenance management and automated deployment features.

The results bring several insights. First, memory capacity at both servers and clients plays a more critical role than the number of vCPUs, given the memory-intensive nature of clustering-based outlier detection executed in the experiments. Configurations using memory-optimized or memory-balanced VMs outperformed compute-optimized ones. Second, clients with less memory became bottlenecks that delayed the aggregation step on the server, increasing workflow makespan and costs. Finally, the analysis of data distribution strategies revealed that while spatial adjacency can improve clustering quality, its benefits are outweighed by the imbalance introduced by variable object densities across the sky regions. Spatially adjacent fragments often became too large, producing bottlenecks in federated outlier detection. On the other hand, random merging achieved better load balance. These findings evidence that while FL is a promising paradigm for astronomical data analysis, its deployment requires careful consideration. Beyond the choice of ML algorithms, resource allocation, workload balancing across clients, and handling uneven data densities are all critical factors for the practicality of FL, especially at the LSST data volume.

As future work, we plan to extend our analysis to include unsupervised learning methods such as DBSCAN for clustering, as well as other techniques for outlier detection. It will also be investigated how to provide adaptive load-balancing, where clients are assigned data volumes proportional to their available resources, as well as hybrid partitioning strategies that consider spatial adjacency and object density. Moreover, although we evaluated the FL workflow in the cloud to prepare for the LSST environment, real-world deployments across IDACs will introduce new challenges.

References

Abbott, T., Abdalla, F., et al. (2018). Dark energy survey year 1 results: cosmological constraints from galaxy clustering and weak lensing. *Physical Review D*, 98(4).

Abbott, T. M. C., Adamów, M., et al. (2021). The dark energy survey data release 2. *The Astrophysical Journal Supplement Series*, 255(2):20.

Aljunaid, S. a. (2025). Secure and transparent banking: Explainable ai-driven federated learning model for financial fraud detection. *J. of Risk and Financial Management*, 18:26.

Alves, A., Carruba, V., et al. (2025). Deep learning identification of asteroids interacting with g-s secular resonances. *Planetary and Space Science*, 258:106062.

Bertin, E. and Arnouts, S. (1996). SExtractor: Software for source extraction. , 117:393–404.

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., de Gusmão, P. P., and Lane, N. D. (2020). Flower: A Friendly Federated Learning Research Framework. *arXiv*.

Casaletto, J., Mackintosh, G., and others. (2022). Using federated learning to overcome data gravity in space. *NASA Technical Reports Server (NTRS)*. Technical Report.

Covey, K. R., Ivezić, Ž., et al. (2007). Stellar SEDs from 0.3 to 2.5 μm: Tracing the Stellar Locus and Searching for Color Outliers in the SDSS and 2MASS. , 134(6):2398–2417.

Cui, L. et al. (2022). Security and privacy-enhanced federated learning for anomaly detection in iot infrastructures. *IEEE Transactions on Industrial Informatics*, 18(5):3492–3500.

D'Isanto, A., Cavuoti, S., et al. (2016). An analysis of feature relevance in the classification of astronomical transients with machine learning methods. *Monthly Notices of the Royal Astronomical Society*, 457(3):3119–3132.

Dong, F., Leung, H., and Drew, S. (2024). Navigating high-degree heterogeneity: Federated learning in aerial and space networks.

Ferreira, W., Kunstmann, L., et al. (2024). Akôflow: um middleware para execução de workflows científicos em múltiplos ambientes conteinerizados. In *XXXIX SBBD*, pages 27–39, Florianópolis/S. SBC.

Fukugita, M., Ichikawa, T., Gunn, J. E., Doi, M., Shimasaku, K., and Schneider, D. P. (1996). The Sloan Digital Sky Survey Photometric System. , 111:1748.

Herschel, M., Diestelkämper, R., and Ben Lahmar, H. (2017). A survey on provenance: What for? what form? what from? *VLDB J.*, 26(6):881–906.

Ivezić, , Kahn, S. M., et al. (2019). Lsst: From science drivers to reference design and anticipated data products. *The Astrophysical Journal*, 873(2):111.

Jurić, M., Kantor, J., Lim, K., Lupton, R. H., et al. (2015). The lsst data management system. *arXiv preprint arXiv:1512.07914*.

Kron, R. G. (1980). Photometry of a complete sample of faint galaxies. , 43:305–325.

Laridi, S. et al. (2024). Enhanced federated anomaly detection through autoencoders using summary statistics-based thresholding. *Scientific Reports*, 14(1):26704.

Lopes, C., Nunes, A. L., Boeres, C., et al. (2023). Provenance-based dynamic fine-tuning of cross-silo federated learning. In *10th CARLA*, volume 1887, pages 113–127. Springer.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations.

Mahabal, A., Sheth, K., Gieseke, F., et al. (2017). Deep-learnt classification of light curves. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8.

McMahan, B., Moore, E., Ramage, D., et al. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS'17*, volume 54, pages 1273–1282.

Mickaelian, A. (2016). Astronomical surveys and big data. *Open Astronomy*, 25(1):75–88.

Nair, D. G., Aswartha Narayana, C. V., et al. (2022). Exploring SVM for Federated Machine Learning Applications. In *Proc. of the ICADCML 2022*, pages 295–305, Singapore.

Razmi, N., Matthiesen, B., Dekorsy, A., and Popovski, P. (2024). Energy-aware federated learning in satellite constellations.

Roth, H. R., Cheng, Y., Wen, Y., et al. (2023). NVIDIA FLARE: federated learning from simulation to real-world. *IEEE Data Eng. Bull.*, 46(1):170–184.

Sattler, F. et al. (2019). Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE TNNLS*, 32:3710–3722.

Savić, D. V., Jankov, I., et al. (2023). The lsst agn data challenge: Selection methods.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.*, 10(2).