Adaptive Toggling of Architectural Patterns for Federated Learning

Luciano Baresi [a], Ivan Compagnucci [b], Livia Lestingi [a], Catia Trubiani [b]

Politecnico di Milano, Milan, Italy [a]
Gran Sasso Science Institute, L'Aquila, Italy [b]

POLITECNICO DI MILANO

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

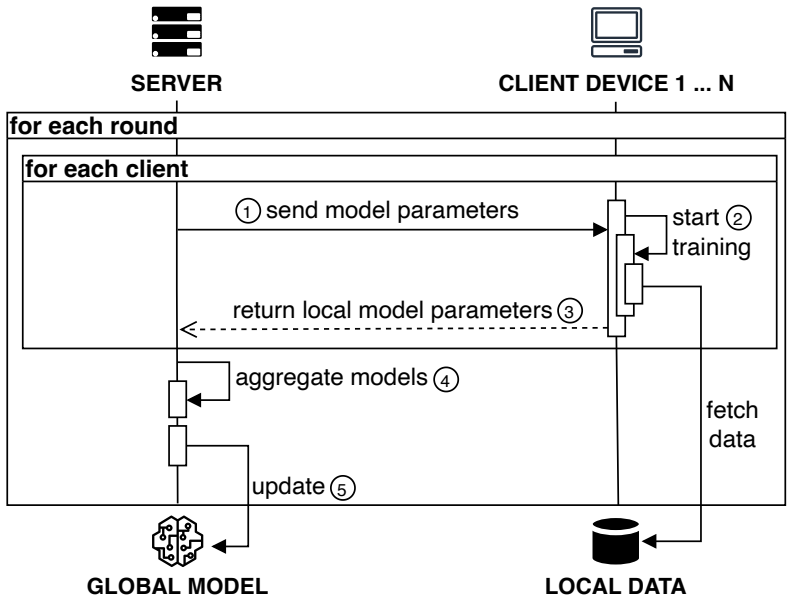
Setting the Scene

Federated Learning

Allows multiple clients to **collaboratively train a global model without sharing the local data** used for training.

Federated Learning round

① Sending Model Parameters
② Local Model Training
③ Sending Trained Model Parameters
④ Model Aggregation



Goal: Train a Global ML Model while **preserving Data Privacy**

This Paper in a Nutshell

State of the Art

- **Architectural Patterns** may improve the performance of FL systems
- `FLRA`: a FL **Reference Architecture** for integrating Architectural Patterns [1]

[1] Lo et al., **"`FLRA`: A Reference Architecture for Federated Learning Systems"**, ECSA 2021.

What is missing?

Architectural Patterns are fixed at design time instead of adapting to evolving system **Performance Metrics** (e.g., accuracy) and **Boundary Conditions** (e.g., data inflow)

Our Contribution

1. Extension of `FLRA` for runtime **Adaptive Architectural Pattern** management
2. Implementation of 3 **Adaptation Policies** for 3 **Architectural Patterns**
3. **Empirical assessment of the framework** under representative FL settings

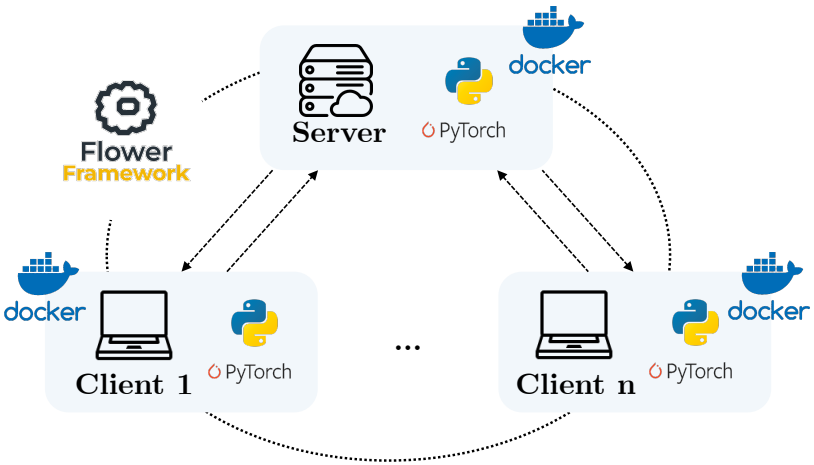
Development Environment

The FLiP Framework

It allows to **dynamically integrate Architectural Patterns and monitor system Performance Metrics** during the Federated Learning process

Main Steps

1. FL System Emulation
2. Docker Environment Setting
3. FL Simulation and Benchmarking



Parameter	Description
$R \in \mathbb{N}$	# of Federated Learning Rounds
C	# of Clients
Res_c	Resources on client C (e.g., CPU, RAM)
D_i^r	Data available on client i at round r
$JSD_i^r \in [0, 1]$	JSD Score of each client i
$\mathcal{T}_i^r \in \mathbb{B}$	Pattern Toggle State for client i at round R
...	...

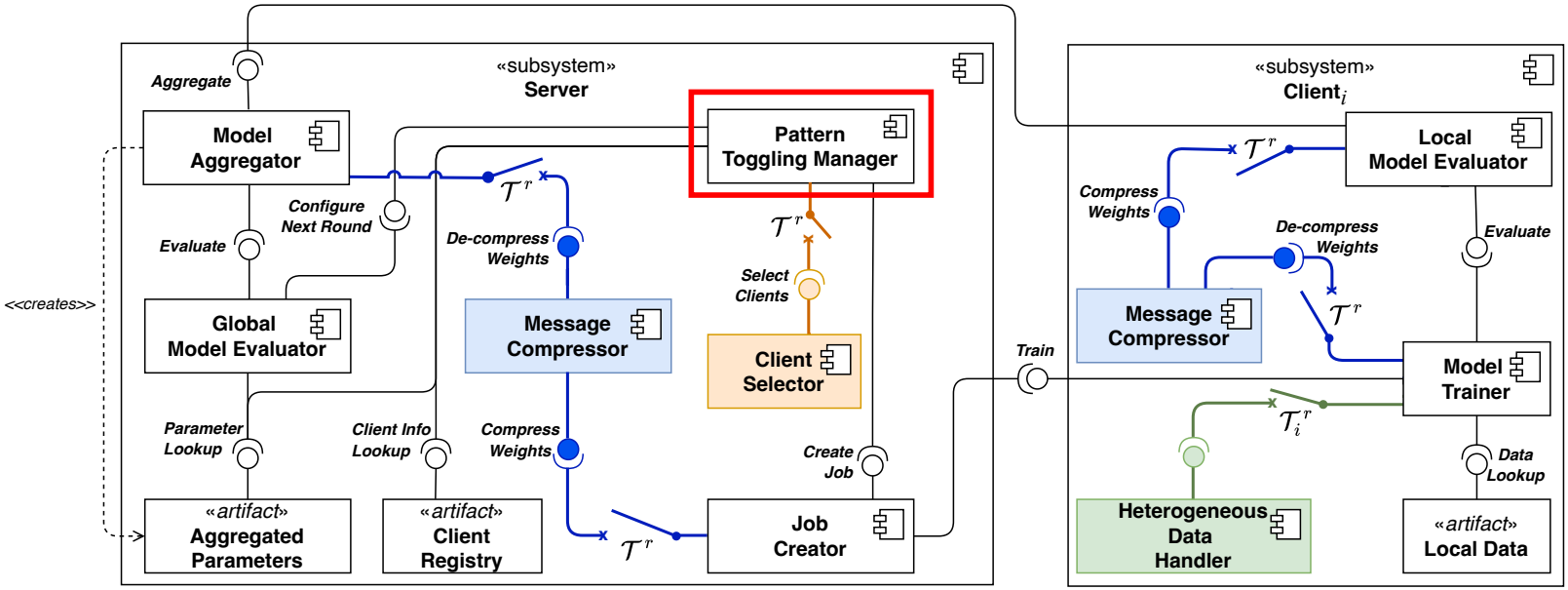
Input Parameters

Parameter	Description
$CT_i^r \in \mathbb{R}_+$	**Communication Time** (Client-Server)
$RT^r \in \mathbb{R}_+$	**Round Time** for each FL Round
$F1^r \in \mathbb{R}_+$	**F1 Score** of the global model

Performance Metrics

Extending the FLRA

- Additions:
  - 3 Architectural Patterns with **Adaptive Switches**
  - **Pattern Toggling Manager** decides if the pattern should be (de)activated
    - ★ **Adaptation Logic** is driven by the **Pattern-specific driver** influenced by evolving performance metrics and boundary conditions

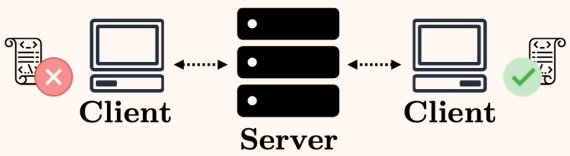


Adaptation Goal

Continuously optimize FL system Performance Metrics by dynamically switching Architectural Patterns at the end of each FL round

Extending FLRA: Architectural Patterns

Client Selector



Selects clients for the next FL round according to specific criteria

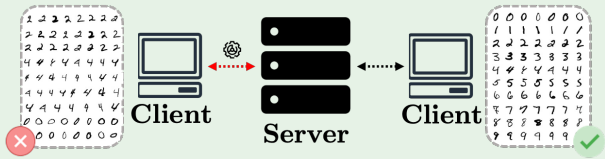
Selection Criteria: # of CPUs

Performance Implications:

- **Pro: Total Round Time** ↓↓
- **Cons: Model Accuracy** ↓↓

  (i.e., Excluding clients may reduce data representativeness, lowering model accuracy)

Heterogeneous Data Handler



Generates synthetic data to mitigate non-IID data distribution across clients

non-IID Client Identification: JSD

Performance Implications:

- **Pro: Model Accuracy** ↑↑
- **Cons: Total Round Time** ↑↑

  (i.e., High computational costs for the generation of high-quality synthetic data)

Message Compressor

Compresses messages exchanged in client-server communications

Compression Algorithm: zlib

Performance Implications:

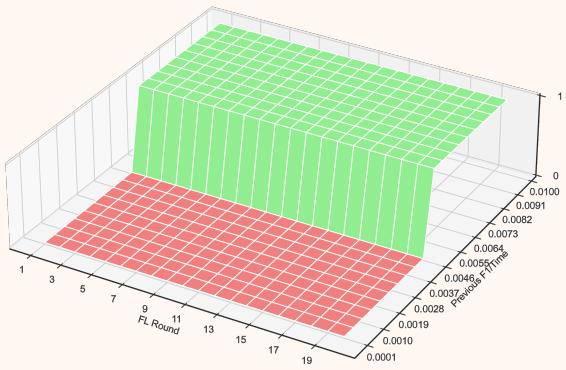
- **Pro: Communication Time** ↓↓
- **Cons: Communication Time** ↑↑

  (i.e., In light-weight models, compression overhead may outweigh the benefits)

Runtime Adaptation of Architectural Patterns

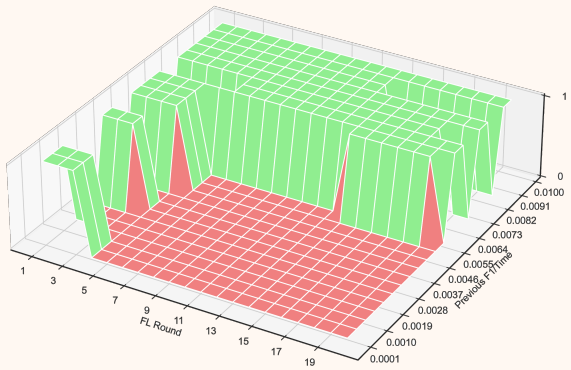
Adaptation Policies: *Which logic drives the adaptation of Architectural Patterns?*

Fixed Rule



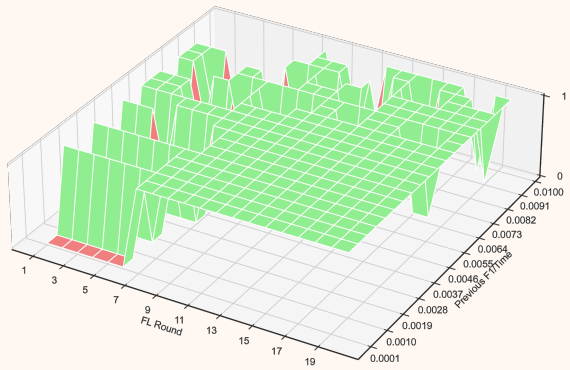
Toggles patterns through **predefined rules** over observed metrics and threshold conditions.

Predictor-based



Uses an **offline-trained predictor** to estimate the impact of activating or deactivating a pattern.

Bayesian Optimization-based



Formulates per-round **pattern toggling as a Bayesian Optimization problem** over the decision space.

Adaptation Drivers: *Which system parameters drive the Architectural Pattern toggling?*

- **Client Selector:** **F1 Score/Total Round Time** + Client Resources (Res_c)
- **Heterogeneous Data Handler:** **F1 Score** + **JSD** (i.e., degree of non-IID data)
- **Message Compressor:** **Communication Time** (i.e., aggregated from Clients)

Evaluation

- **RQ1.** How **effective** is FLiP?
- **RQ2.** What is the **offline** overhead introduced by FLiP?
- **RQ3.** What is the **online** overhead introduced by FLiP?

Experiment Design

- Emulation rather than Simulation
- Different ML Scenarios
- Statistical Tests (i.e., Mann-Whitney U test and Vargha-Delaney)
- Runtime changing boundary conditions

Feature	Value
Model under training	{MLP, CNN}
Training dataset	{AG_NEWS, CIFAR-10}
FL rounds	20
Memory limit per client	4GB
CPUs per low-spec client	1
CPUs per high-spec client	2
N. high-spec clients	{2, 3, 4, 5, 8, 10}
N. low-spec clients	{2, 3, 4, 5, 8, 10}
Data distribution type	{IID, non-IID}
Client data inflow	{one-shot, batched}
Network condition	{stable, unstable}

Evaluation Subjects Features

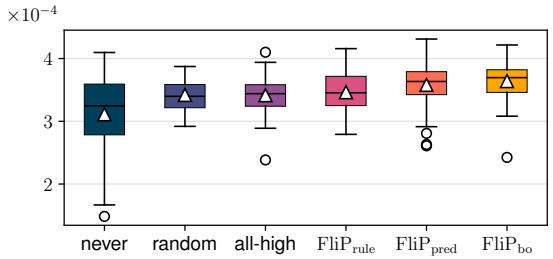
FLiP Effectiveness

- Architectural Pattern are evaluated **in isolation**
- FLiP is compared against 3 **static baselines**
- Effectiveness is measured through **Pattern-specific metrics** derived from theoretical trade-offs:

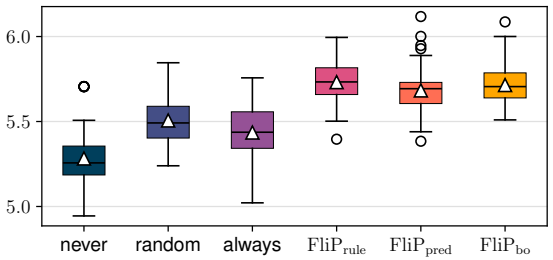
$$y_1(R) = \frac{F1_R}{\sum_{r=1}^{R} RT_r} \qquad \textbf{Client Selector}$$

$$y_2(R) = \sum_{r=1}^{R} F1_r \qquad \textbf{Heterogeneous-Data-Handler}$$

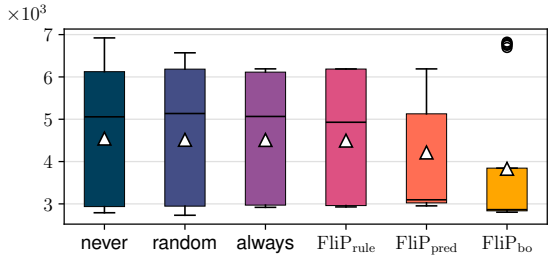
$$y_3(R) = \sum_{r=1}^{R} \sum_{i=1}^{|C|} CT_i^r \qquad \textbf{Message Compressor}$$



$y_1(R)$ – Client Selector



$y_2(R)$ – Heterogeneous-Data-Handler



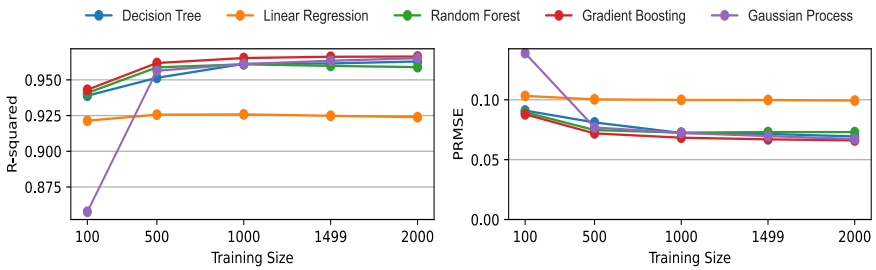
$y_3(R)$ – Message Compressor

RQ1. How effective is FLiP?

Adaptation improves FL with gains of up to **10% in target performance metrics** (i.e., y_1, y_2, y_3) compared to baselines

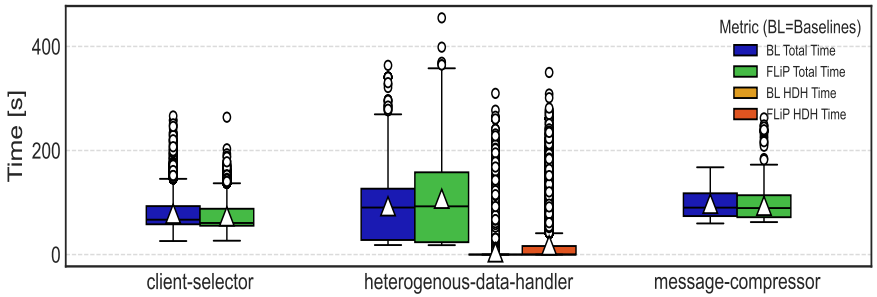
FLiP Overhead

- **Offline Overhead:** Time and Data required to build the **Adaptation Policy**
  - Required **only once before deployment**
  - Policies trained from **past FL executions**



RQ2 – Offline overhead (Image Classification)

- **Online Overhead:** Time required for **applying adaptation** on the fly
  - Required **at the end of each FL round**
  - Limited to the runtime **adaptation decisions**



RQ3 – Online overhead (Image Classification)

RQ2-3. What is the Offline/Online time overhead of FLiP?

Runtime adaptation needs **moderate Offline costs** (i.e., a couple of hours for training the model's adaptation policies) and **negligible Online overhead** (i.e., +27% round time for Heterogeneous-Data-Handler, near-zero for Client-Selector and Message-Compressor)

Conclusion & Future Work

Takeaway Message

To maximize the benefits of **Architectural Patterns** in Federated Learning, they should be **adaptively (de)activated** based on evolving system performance metrics and boundary conditions

Future Work

- Runtime adaptation of **Combined Architectural Patterns**
- Implement a new **Adaptation Policy without offline costs**
- Experiment with **Additional ML Tasks, Models, and Dataset**

Thank you!

Adaptive Toggling of Architectural Patterns for Federated Learning

Luciano Baresi [a], Ivan Compagnucci [b], Livia Lestingi [a], Catia Trubiani [b]

Politecnico di Milano, Milan, Italy [a]
Gran Sasso Science Institute, L'Aquila, Italy [b]

POLITECNICO DI MILANO

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore