Towards AI Agents for Selecting Architectural Patterns in Federated Learning Systems

Ivan Compagnucci[1,*,†], Catia Trubiani[1,*,†]

[1]Gran Sasso Science Institute, L'Aquila, Italy

Abstract

Recent studies have shown that the adoption of architectural patterns in Federated Learning systems can lead to significant advantages in terms of system efficiency (e.g., improved model accuracy and faster training time). However, identifying which architectural patterns or combinations of them can lead to improvements in a given system remains a manual and error-prone process. To address this challenge, we take an initial step toward enabling the automated (de)selection of architectural patterns in Federated Learning systems. We extend our benchmarking platform, namely AP4Fed, with the conceptualization of AI Agents that are in charge of reasoning about possible architectural alternatives. At the beginning of each Federated Learning round, one or multiple AI agents analyze current system metrics (e.g., model accuracy, total round time) and contextual information (e.g., resource capabilities of participating clients), collaboratively deciding which architectural patterns should be (de)activated. This approach can support software architects by introducing an AI-driven decision-making mechanism that dynamically selects architectural patterns during Federated Learning simulations, aiming to improve the system efficiency and to reduce the manual tuning effort.

Keywords

Federated Learning, Architectural Patterns, AI Agent, Performance Analysis

1. Introduction

Federated Learning is a distributed machine-learning paradigm mainly introduced for data privacy, since multiple clients collaboratively train a single global model without exposing their raw data [1, 2]. Instead of sharing sensitive data used for the training, each client performs local learning on its data and transmits only model updates (i.e., trained model hyper-parameters) to the server [1, 3]. Recent studies on Federated Learning [1**?** , 4] identify *performance optimization* as a key challenge in this domain.

Our previous work [5] has been devoted to investigate the performance of Federated Learning systems while adopting a set of domain-specific architectural patterns [2]. We developed a benchmarking platform, namely AP4Fed, for systematically assessing the impact of each pattern on Federated Learning system performance. AP4Fed allows to perform custom Federated Learning simulations by emulating system setups (a server and n clients with configurable CPU allocations and Federated Learning rounds), plugging in architectural patterns (e.g., a *Client Selector* to include/exclude clients according to specific criteria, such as computational power), and automatically extracting performance metrics (e.g., model accuracy, total round time). However, the optimal (de)selection of architectural patterns, given a specific simulation system setting, is still an open problem.

Recent advancements in the field of Artificial Intelligence (AI) have shown the effectiveness of intelligent agents in software systems to support decision-making, adaptation, and automation [6]. AI-Agents are increasingly integrated into complex software architectures to monitor system behavior, learn from past outcomes, and suggest real-time interventions. They have proven useful in dynamic environments where conditions and objectives may change frequently, enabling systems to self-optimize

QualITA 2025: The Fourth Conference on System and Service Quality, June 25 and 27, 2025, Catania, Italy

*Corresponding author.

†These authors contributed equally.

ivan.compagnucci@gssi.it (I. Compagnucci); catia.trubiani@gssi.it (C. Trubiani)

https://ivancomp.github.io (I. Compagnucci); https://cs.gssi.it/catia.trubiani (C. Trubiani)

0000-0002-1991-0579 (I. Compagnucci); 0000-0002-7675-6942 (C. Trubiani)

without manual intervention [6]. These premises encourage the adoption of AI in the context of selecting architectural patterns while optimizing the performance of Federated Learning systems.

The goal of this paper is to propose a conceptual framework for automating the (de)selection of architectural patterns in Federated Learning systems, thus enabling dynamic adaptation based on real-time system performance and contextual information. To this end, we foresee two possible AI-based solutions: (i) a Single-AI-Agent that centrally reasons over all pattern selections, and (ii) a Multi-AI-Agent system where specialized agents independently manage individual patterns under a coordinating AI Agent (Manager). At the end of each Federated Learning round, the agent(s) analyze contextual data and past system performance metrics to decide which architectural patterns are candidate to be (de)activated. This approach aims to maximize overall system performance, while mitigating the combinatorial explosion of possible pattern configurations and relieving software architects from manual and trial-and-error tuning.

2. Preliminaries

2.1. AP4Fed: A Federated Learning Benchmarking Platform

AP4Fed [1] is a Federated Learning benchmarking platform built on top of Flower [7], i.e., a Python library designed to perform Federated Learning simulations. AP4Fed extends Flower by integrating architectural patterns and monitoring capabilities to enable performance analysis of Federated Learning simulations. The tool uses Docker-Compose to emulate a realistic Federated Learning setup, with one container as the central server and multiple containers as clients performing local training and sending updates for aggregation.

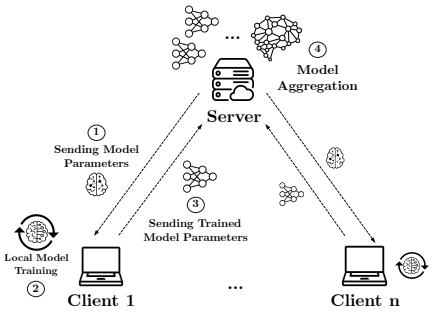


Figure 1: Federated Learning Overview.

	Parameter	Description
Input Parameters	NUM_ROUNDS	Number of FL rounds
	nC	Client container instances
	n_CPU	CPUs per Client
	RAM	RAM per Client
	AP_List	List of Architectural Patterns
Evaluation Metrics	Training Time	Local training time
	Communication Time	Client–Server Communication time
	Total Round Time	Total time per FL round
	Model Accuracy	Aggregated model accuracy

Table 1: Input Parameters and Evaluation Metrics.

Federated Learning Simulation. The main steps of a Federated Learning simulation are depicted in Figure 1. It begins with a central server broadcasting the initial global model parameters (e.g., model weights) to all participating clients ①. Each client trains the model locally using its local data ② and sends the updated weights back to the server ③. The server aggregates these updates to produce a new global model ④, which is redistributed to the clients for the next training round. This cycle represents a single round of Federated Learning and can be repeated multiple times until the model converges.

Table 1 summarizes the *Input Parameters* and *Evaluation Metrics* used in AP4Fed. These values are stored in a JSON file generated via the GUI provided by AP4Fed. This information is accessed at the beginning of each round to initialize the system's parameters. The *Input Parameters* include NUM_ROUNDS (number of Federated Learning rounds), nC (number of client), n_CPU and RAM per client, and the list of architectural patterns (AP_List) enabled during the simulation. The *Evaluation Metrics* capture system performance. Training Time refers to the duration of client local training, Communication Time measures the time spent exchanging data between clients and server, Total Round Time covers the overall duration per round, and Model Accuracy reflects the accuracy of the global model.

[1] https://github.com/IvanComp/AP4Fed

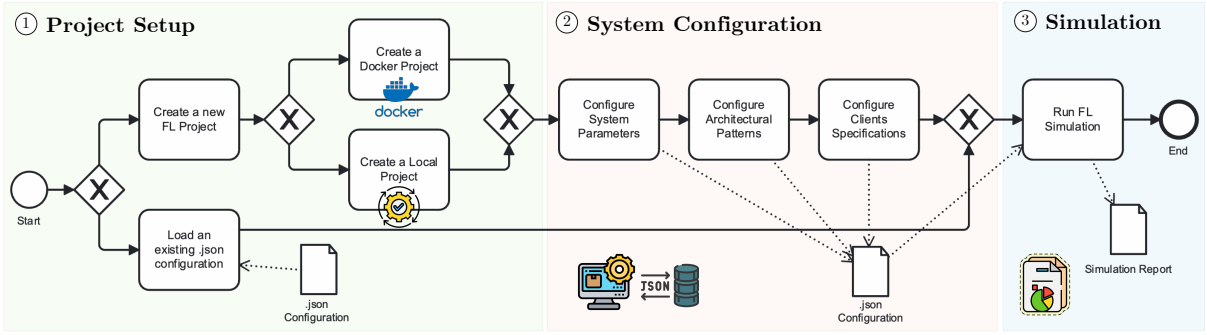


Figure 2: Overview of the Federated Learning Simulation Process with AP4Fed.

AP4Fed Federated Learning Simulation. Figure 2 shows a BPMN diagram outlining the workflow of using AP4Fed to run an Federated Learning simulation. The process is structured into three key phases. The first step, *Project Setup*, initializes a new Federated Learning simulation. A GUI allows users to either create a new setup or load a pre-configured JSON file, supporting both customization and portability. Simulations can run in a local environment, which is ideal for rapid testing of ML strategies, or in a container-based setup that better emulates real-world conditions while managing resources efficiently and avoiding overcommitment [8].

In the second step, *System Configuration*, the GUI allows users to specify the input parameters (see Table 1) that populate the configuration JSON file. These parameters include general simulation settings, the computational resources allocated to each client, and the set of architectural patterns to be implemented during the simulation. In the final phase, *Simulation*, the system executes the Federated Learning simulation considering the *Input Parameters* defined by the user. At the end of the simulation AP4Fed provides a detailed report containing the evaluation metrics to assess the system's performance.

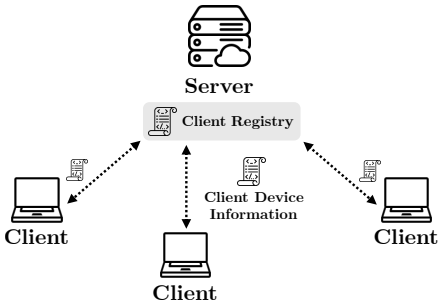
2.2. AP4Fed: Architectural Patterns

Architectural patterns represent reusable solutions to common design problems in complex systems [9]. Lo et a. [2] present a collection of architectural patterns tailored for Federated Learning systems. These patterns cover client management (i.e., managing client devices), model management (i.e., managing the aggregated model), model training (i.e., managing the training phase), and model aggregation (i.e., managing the server aggregation phase). The current version of AP4Fed supports four architectural patterns from Lo et a. [2], which are described in the following.

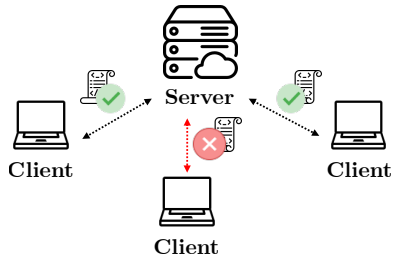
Client Registry. As shown in Figure 3a, this pattern introduces a centralized registry that stores key attributes of each client, such as unique identifiers, available computational resources (e.g., number of CPU cores), and other relevant information [2]. The server leverages this registry to support the simulation by performing tasks such as client sampling or collecting data for evaluation metrics. This pattern serves a coordination purpose and has no direct impact on the system's performance.

Client Selector. The Client Selector pattern introduces a mechanism for selecting a subset of client devices to participate in the Federated Learning round according to specific criteria [10, 2]. As depicted in Figure 3b, the server evaluates each client by analyzing the information stored in the Client Registry, to exclude or include clients based on selection criteria. Such criteria can be categorized as follows: (i) *resource-based* factors, which evaluate the computational capabilities of devices; (ii) *data-based* factors, which assess the quality, heterogeneity, and volume of the data each client holds; and (iii) *performance-based* factors, which assess each client's contributions to the global model enhancement based on their performance in the latest round [10]. This assessment evaluates each client's characteristics to determine whether and which ones are best suited to contribute to the global model aggregation.

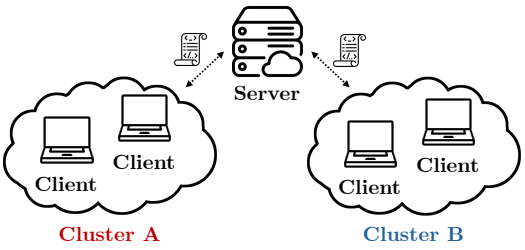
Client Cluster. This pattern groups clients into homogeneous clusters based on the similarity of their characteristics, with clusters defined by computational resources (for example processing power and memory capacity), network capabilities (i.e., meaning bandwidth and connection reliability) and data



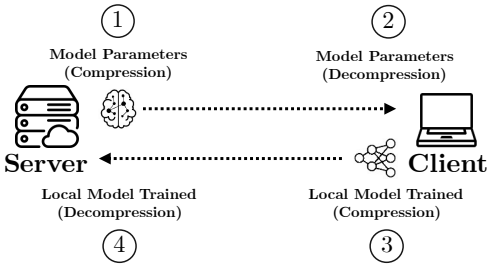
(a) Client Registry Architectural Pattern.



(b) Client Selector Architectural Pattern.



(c) Client Cluster Architectural Pattern.



(d) Message Compressor Architectural Pattern.

Figure 3: Architectural Patterns implemented in AP4Fed.

partitioning type (i.e. how the dataset is split and distributed among devices). As shown in Figure 3c, the server leverages these client clusters to deploy customized aggregation strategies for each group, improving overall system efficiency (i.e., addressing variations in data distribution and managing heterogeneous device capabilities).

Message Compressor. This pattern aims to cuts communication overhead in Federated Learning by applying the `zlib` (de)compression algorithm [11] at both server and client end points. As shown in Figure 3d, The server first compresses model parameters before sending them to clients, which decompress the data for local training. After updating the model, clients recompress their parameters and transmit them back, where the server decompresses and aggregates the results. This bidirectional compression cycle reduces data volume exchanged in each training round, reducing the communication time and conserving bandwidth, which is may be beneficial for clients with limited network capacity [12].

2.3. AI Agents in Software Systems

An AI agent is a system designed to autonomously perform complex tasks going beyond traditional software automation by making intelligent decisions, e.g., adapting to context, and optimizing the execution of workflows [6]. AI Agents mark a new era in workflow automation: they can handle uncertain situations by reasoning over available *input* data and generating *output* in the form of suggestions to support decision making. Unlike conventional automation, AI Agents are suited to workflows where deterministic and rule-based approaches fall short, due to shifting context and the need to reason over many complex situations involving multiple factors [6].

Engineering Single-AI-Agent Systems. Figure 4 depicts an overview of a Single-AI-Agent architecture [13]. Specifically, the main components are: (*i*) a *Reasoning Model* which is the "brain" of the AI Agent, implemented with a pre-trained LLM such as `GPT-o3` [6]. It ingests an input (e.g., contextual data) and produces an output (e.g., a reasoned plan) by leveraging its large-scale knowledge and reasoning capabilities; (*ii*) the *Instructions & Guardrails*, i.e., a static prompt provided to the agent to guide the agent's reasoning, define its goal, and constrain the agent's behavior while preventing malicious or inappropriate outputs. These guardrails ensure that the agent's decisions remain aligned with desired objectives and ethical considerations. Finally, (*iii*) *Tool(s)* serve as interfaces that enable

interaction with the external environment and the AI Agent itself, allowing the agent to acquire data as input, process it internally, and subsequently issue decisions as output.

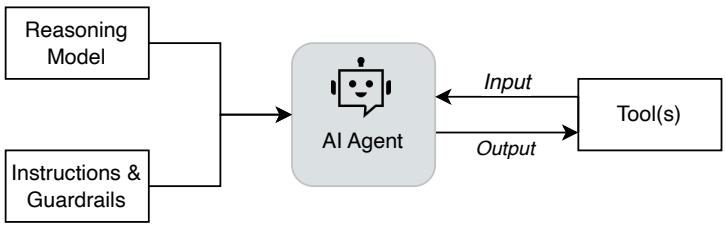


Figure 4: Overview of a Single-AI-Agent Architecture (adapted from [13]).

Engineering Multi-AI-Agent Systems. Multi-AI-Agent architectures are either *centralized*, where a single AI Agent (i.e., manager) orchestrates a set of specialized sub-agents, or *decentralized*, where peer agents delegate tasks among themselves [13]. In this work, we adopt the centralized design, as illustrated in Figure 5. In contrast to the Single-AI-Agent architecture, where a single LLM handles all reasoning tasks, the Multi-AI-Agent paradigm employs a collection of pre-trained LLMs, each specialized and sized according to the complexity and performance needs of its specific sub-task. These agents operate under the coordination of a central manager agent, which oversees their interactions and integrates their outputs to make coherent overall decisions. *Instructions & Guardrails* are decentralized: rather than a single prompt and uniform policy, each agent is provisioned with its own prompt tailored to its specific task. Finally, the *Tool* communicates with the centralized AI-Agent Manager which serves as interface enabling the agents to gather relevant input data, internally process and analyze this information, and subsequently generate the output.

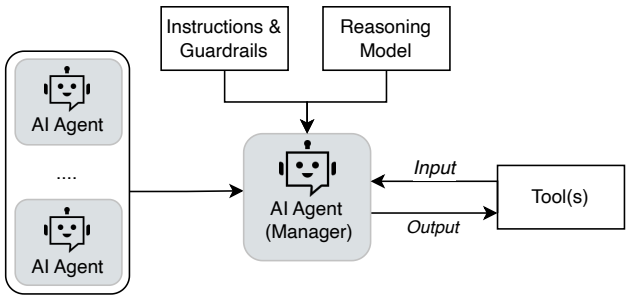


Figure 5: Overview of a Multiple-AI-Agent (Centralized) Architecture (adapted from [13]).

3. Conceptualization

In this section, we propose two strategies to extend AP4Fed with AI agents that monitor and reason on real-time system performance and contextual information, enabling the dynamic (de)selection of architectural patterns during Federated Learning simulations. This implies both gains and pains that we discuss in the following.

3.1. AI Agents in Federated Learning: Overview

Let us consider the goal of pursuing the *performance optimization* in Federated Learning systems. A traditional static configuration of architectural patterns acts like a fixed checklist, applying the same setup each Federated Learning round regardless of context. In contrast, an AI-based agent acts like

an "intelligent" system architect, analyzing past performance metrics and dynamically (de)selecting patterns at each round based on the evolving system state. This possibility to contextually reason and adjust to changing conditions is precisely what enables such an agent to manage complex and variable environments, thus continuously optimizing the Federated Learning system performance that represents our goal.

3.2. Engineering the Single-AI Architecture for Architectural Pattern (De)Selection

Figure 6 depicts an overview of the proposed strategy. The AI Agent is based on a Single-AI-Agent architecture [6], which includes the three core components described in Section 2.3.

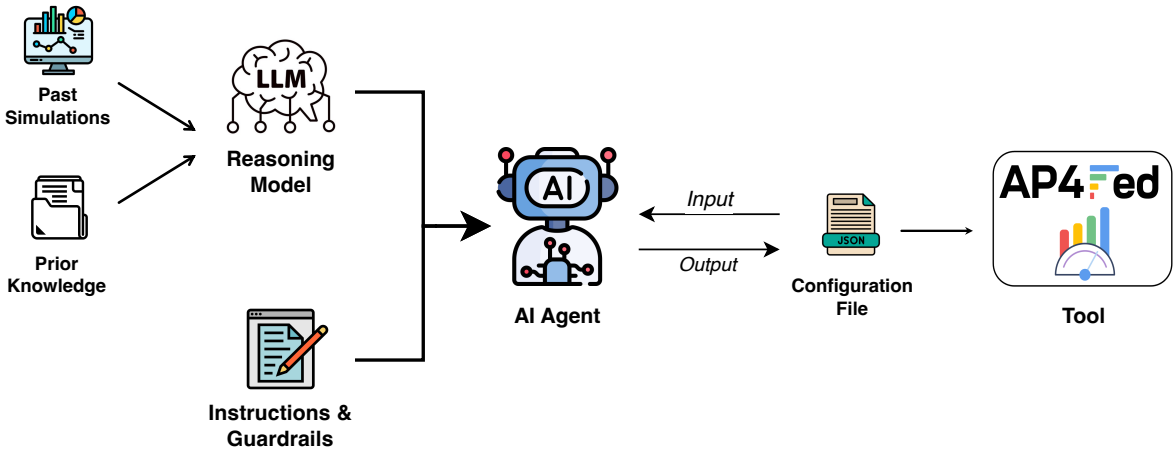


Figure 6: Overview of the Proposed Strategy with a Single-AI Architecture.

The *Reasoning Model* is in charge of interpreting the *Input Parameters* and *Evaluation Metrics* of the Federated Learning system. This information is extracted from the JSON configuration file (please refer to Table 1). As support for the interpretation of such data, we fine-tune the reasoning model with (i) past simulations (i.e., raw data) and (ii) prior knowledge (i.e., elaborated data that report the findings from our performance analysis of architectural patterns [5]). Based on its reasoning, the agent updates the list of architectural patterns (e.g., the message compressor is set to be disabled), which are then used by AP4Fed to execute the next round accordingly. The agent's behavior is guided by a set of *Instructions & Guardrails*, i.e., a static prompt that defines the goal to achieve.

3.3. Engineering the Multiple-AI-Agent for Architectural Pattern (De)Selection

In this strategy, the selection of architectural patterns is managed by multiple independent AI agents, each trained in deciding whether to (de)activate a specific architectural pattern. Unlike a Single-AI-Agent architecture, which uses a single reasoning model to manage all architectural pattern selections, the Multiple-AI-Agent system decomposes the decision-making process into several agents coordinated by a central AI agent manager. Each sub-agent receives input data such as current system metrics (e.g., model accuracy, total round time), contextual information (e.g., client specifications), past simulation data, and prior knowledge related to its assigned pattern. For example, the Client Selector agent analyzes resource availability, data quality, and client performance to decide which clients to include in the training round. The Client Cluster agent reasons about grouping clients with similar characteristics to optimize aggregation strategies. The Message Compressor agent evaluates communication overhead and the global model size to determine if compression should be enabled. Each agent is guided by dedicated instructions and guardrails tailored to its specific task. The central manager agent coordinates each "pattern" agents, integrating their individual decisions into a unified configuration of architectural patterns to (de)activate in the upcoming Federated Learning round.

Figure 7 depicts the architecture of the Multi-AI-Agent strategy, highlighting the separation of concerns among agents and the flow of inputs and outputs within the system. This design aims to mitigate the combinatorial complexity of simultaneously managing multiple architectural patterns by distributing the reasoning load and enabling fine-grained control over each pattern.

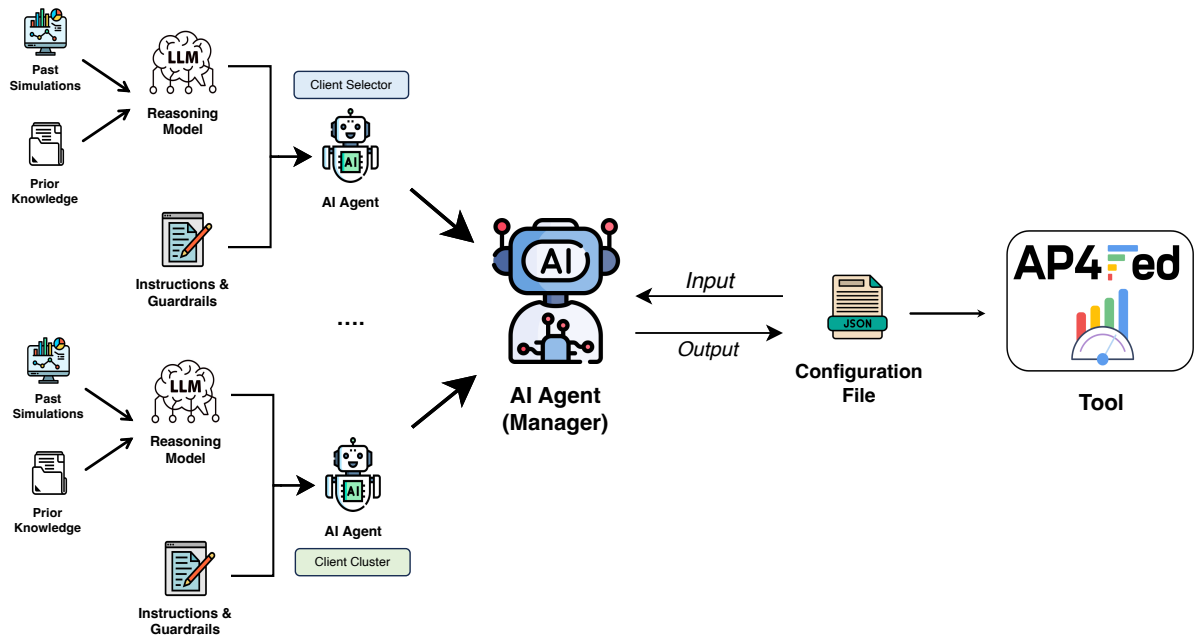


Figure 7: Overview of the Proposed Strategy with a Multi-AI-Agent Architecture.

3.4. Expected Results and Open Challenges

The approaches proposed in this paper paves the way for the run-time selection of architectural patterns while reasoning on the current system parameters and aiming to optimize the performance of Federated Learning systems. To this end, the inclusion of an AI agent is of key relevance to enable intelligent and context-aware reasoning, thus providing automated suggestions to (de)activate architectural patterns while attempting to optimize Federated Learning system performance. However, their introduction triggers new issues and challenges that we summarize in the following, with the perspective of being addressed as part of our future research.

First, designing effective decision-making for the AI agent is challenging, as it must adapt to dynamic conditions and multiple parameters. This requires suitable reward functions and efficient fine-tuning strategies [6]. Furthermore, AI Agents relies heavily on data to function effectively. The more accurate, diverse, and large is the dataset, the better the AI will perform [6]. In our context, this means to feed the reasoning model with proper prior knowledge and past simulations, thus improving the outcome of the decision-making process. Second, the computational overhead introduced by the AI agent itself may be significant, so minimizing its impact is desirable [6].

4. Related Work

Architectural Solutions in Federated Learning. Several prior works propose architectural solutions or investigate approaches aimed at optimizing and evaluating the performance of Federated Learning systems from a software architecture perspective. FLRA [14] proposes a pattern-oriented reference architecture for Federated Learning systems, providing an end-to-end blueprint covering stages such as job creation, model deployment, and monitoring. This architecture combines insights from academic research and industrial best practices. Zhang et al. [15] compare four Federated Learning system

architectures: centralized, hierarchical, regional, and decentralized. They evaluate these architectures based on communication overhead, model convergence speed, and scalability. Lo et al.[2] and Di Martino et al.[16] provide catalogs of architectural patterns tailored specifically for Federated Learning, with Di Martino et al. focusing on healthcare applications. Among these patterns, the Client Selector, also mentioned in our work, is a notable example. Each pattern corresponds to a specific phase in the Federated Learning model lifecycle, providing practitioners with concrete design strategies.

Many studies benchmark Federated Learning performance under diverse conditions. Li et al. [17] provide a comprehensive survey addressing how Federated Learning optimization must consider constraints such as limited device resources, non-IID data distributions, and privacy requirements. Lai et al. [18] introduce FedScale, a benchmarking suite featuring realistic datasets and a scalable runtime to promote reproducible Federated Learning research. FedScale supplies high-level APIs for straightforward implementation, deployment, and evaluation of Federated Learning algorithms across heterogeneous hardware and software environments. Other works focus on client selection strategies [19, 20], discussing principles like data heterogeneity and hardware limitations, alongside scheduling challenges and open research questions. Notably, Compagnucci et al. [5] select, implement, and quantitatively compare four architectural patterns from Lo et al. [2], revealing important trade-offs between performance gains and computational costs.

Self-Adaptation in Federated Learning. To the best of our knowledge, Aljohani et al. [21] present the first survey specifically dedicated to self-adaptation techniques within Federated Learning systems. The authors explore the integration of Self-Adaptive Systems, emphasizing the necessity of feedback control mechanisms, such as the MAPE-K loop, to dynamically manage resource constraints, model configurations, and client participation. While existing studies typically employ various self-adaptation techniques, none have yet employed AI agents for dynamically (de)selecting architectural patterns within Federated Learning systems.

In the following we discuss practical approaches that have been proposed for enabling self-adaptation within Federated Learning. Baresi et al. [22] explicitly frame Federated Learning as a self-adaptive problem, aiming to dynamically manage training configurations to improve performance under resource constraints. Authors employ interpolation techniques using accuracy measurements from previous rounds to estimate optimal training effort. Wang et al. [23] also focus on adaptivity in FL, but in the context of mobile edge computing. They propose a control algorithm that dynamically adjusts the frequency of global aggregations based on system dynamics, data distribution, and resource budgets. Their method minimizes the training loss under fixed resource constraints and is supported by a theoretical convergence analysis that considers non-i.i.d. data distributions. Nishio and Yonetani [24] address the efficiency challenges in Federated Learning introducing a dynamic client selection strategy based on available client resources, including computational capabilities and wireless connection quality. This approach begins with a preliminary phase where the MEC server collects resource information from the clients, allowing it to select the optimal subset of clients that can effectively contribute to the training process within specified deadlines. In [25], also address self-adaptation in Federated Learning by proposing dynamic sampling and selective masking strategies to improve communication efficiency. Their approach dynamically adjusts the fraction of selected client models and selectively masks model parameters based on their importance. This adaptive mechanism helps manage communication loads efficiently, resulting in significant performance improvements.

In summary, foundational architectural studies in Federated Learning [14, 2] tend to emphasize qualitative analysis combining literature review and industry practices. Although some early-stage approaches for self-adaptation in Federated Learning exist [21], none have employed AI agents specifically to adapt architectural decisions regarding the selection and integration of FL patterns.

5. Conclusion

This paper propose two conceptual strategies to automate the (de)selection of architectural patterns in Federated Learning systems through AI Agents. We propose both Single-AI-Agent and Multi-AI-

Agent architectures that dynamically reason on real-time performance metrics and contextual data to (de)activate architectural patterns at each round. While the single agent provides a centralized reasoning mechanism, the multi-agent approach decomposes decision-making into specialized sub-agents coordinated by a central manager. The proposed methodologies pave the way for enhanced system adaptability and performance optimization, reducing manual tuning efforts. However, they also introduce new challenges, including the design of effective coordination strategies, managing the computational overhead of multiple agents, and ensuring timely decisions within tight Federated Learning round constraints.

As future work, we plan to implement and evaluate both the Single-AI-Agent and Multiple-AI-Agent architectures. This will enable us to compare their performance and effectiveness, thereby determining which strategy better supports the dynamic architectural pattern selection in Federated Learning. Moreover, we plan to measure the computational overhead introduced by the AI agents, ensuring that the benefits of the approach outweigh the introduced computational costs.

Acknowledgments

This work has been partially funded by the MUR-PRIN project 20228FT78M DREAM (modular software Design to Reduce uncertainty in Ethics-based cyber-physicAl systeMs), MUR Department of Excellence 2023 - 2027 for GSSI, and PNRR ECS00000041 VITALITY.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools. During the preparation of this work, the author(s) used ChatGPT-5 and Grammarly in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A Survey on Federated Learning, Knowledge-Based System 216 (2021) 106775.

[2] S. K. Lo, Q. Lu, L. Zhu, H.-Y. Paik, X. Xu, C. Wang, Architectural Patterns for the Design of Federated Learning Systems, Journal of Systems and Software 191 (2022) 111357.

[3] Q. Li, Y. Diao, Q. Chen, B. He, Federated Learning on Non-IID Data Silos: An Experimental Study, International Conference on Data Engineering (ICDE) (2021) 965–978.

[4] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, D. Dou, From Distributed Machine Learning to Federated Learning: A Survey, Knowledge and Information Systems 64 (2022) 885–917.

[5] I. Compagnucci, R. Pinciroli, C. Trubiani, Performance Analysis of Architectural Patterns for Federated Learning Systems, in: International Conference on Software Architecture, ICSA, IEEE, 2025, pp. 289–300.

[6] L. Bass, Q. Lu, I. Weber, L. Zhu, Engineering AI Systems: Architecture and DevOps Essentials, Addison-Wesley Professional, 2025.

[7] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, N. D. Lane, Flower: A Friendly Federated Learning Research Framework, CoRR abs/2007.14390 (2020).

[8] M. C. Cohen, P. W. Keller, V. S. Mirrokni, M. Zadimoghaddam, Overcommitment in Cloud Services: Bin Packing with Chance Constraints, Management Science 65 (2019) 3255–3271.

[9] M. Richards, Software Architecture Patterns, volume 4, 2015.

[10] C. Briggs, Z. Fan, P. Andras, Federated Learning with Hierarchical Clustering of Local Updates to Improve Training on Non-IID Data, in: International Conference on Neural Network, 2020, pp. 1–9.

[11] M. Adler, J.-L. Gailly, zlib: A Data Compression Library, 2024. URL: https://github.com/madler/zlib.

[12] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, M. Jirstrand, A Performance Evaluation of Federated Learning Algorithms, in: International Workshop on Distributed Infrastructures for Deep Learning (DIDL), 2018, pp. 1–8.
[13] OpenAI, A Practical Guide to Building Agents, Technical Report, OpenAI, Inc., 2025.
[14] S. K. Lo, Q. Lu, H.-Y. Paik, L. Zhu, FLRA: A reference architecture for federated learning systems, in: European Conference on Software Architecture, Springer, 2021, pp. 83–98.
[15] H. Zhang, J. Bosch, H. H. Olsson, Federated learning systems: Architecture alternatives, in: Asia-Pacific Software Engineering Conference (APSEC), IEEE, 2020, pp. 385–394.
[16] B. Di Martino, D. Di Sivo, A. Esposito, Architectural patterns for software design problem-solving in the implementation of federated learning structures within the e-health sector, in: International Conference on Advanced Information Networking and Applications, Springer, 2024, pp. 347–356.
[17] T. Li, A. K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE signal processing magazine 37 (2020) 50–60.
[18] F. Lai, Y. Dai, S. Singapuram, J. Liu, X. Zhu, H. Madhyastha, M. Chowdhury, FedScale: Benchmarking Model and System Performance of Federated Learning at Scale, in: Proceedings of Machine Learning Research (PMLR), 2022, pp. 11814–11827.
[19] L. Fu, H. Zhang, G. Gao, M. Zhang, X. Liu, Client selection in federated learning: Principles, challenges, and opportunities, IEEE Internet of Things Journal 10 (2023) 21811–21819.
[20] S. Mayhoub, T. M. Shami, A review of client selection methods in federated learning, Archives of Computational Methods in Engineering 31 (2024) 1129–1152.
[21] A. Aljohani, O. Rana, C. Perera, Self-adaptive federated learning in internet of things systems: A review, ACM Computing Surveys 57 (2025) 1–36.
[22] L. Baresi, G. Quattrocchi, N. Rasi, Federated machine learning as a self-adaptive problem, in: International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE, 2021, pp. 41–47.
[23] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, K. Chan, Adaptive federated learning in resource constrained edge computing systems, IEEE J. Sel. Areas Commun. 37 (2019) 1205–1221.
[24] T. Nishio, R. Yonetani, Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge, in: International Conference on Communications, ICC, IEEE, 2019, pp. 1–7.
[25] S. Ji, W. Jiang, A. Walid, X. Li, Dynamic Sampling and Selective Masking for Communication-Efficient Federated Learning, IEEE Intell. Syst. 37 (2022) 27–34.