Towards AI Agent 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 a first step toward enabling the automated selection of architectural patterns in Federated Learning systems. We propose a conceptual extension of our Federated Learning benchmarking platform, namely AP4Fed, by integrating an artificial intelligence (AI)-based Agent capable of reasoning about architectural decisions. At the beginning of each Federated Learning round, the agent analyzes current system metrics (e.g., model accuracy, total round time) along with contextual system information (e.g., resource capabilities of clients participating to the learning), thus determining 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 (FL) 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 own data and transmits only model updates (i.e., trained model hyper-parameters) to the server [1]. Recent studies on FL [1, 3] identify *performance optimization* as a key challenge in this domain.

Our previous work [4] has been devoted to investigate the performance of FL 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 FL system performance. AP4Fed allows to perform custom FL simulations by emulating system setups (a server and n clients with configurable CPU allocations and FL 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 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 [5]. 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 without manual intervention [5]. These premises encourage the adoption of AI in the context of selecting architectural patterns while optimizing the performance of FL systems.

QualITA'25: The Fourth Conference on System and Service Quality, June 25–26, 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)

The goal of this paper is to explore the definition of a conceptual approach for fully automating the selection of architectural patterns. To this end, we foresee the introduction of an AI-based agent that dynamically coordinates the inclusion or exclusion of architectural patterns in FL simulations. At the end of each FL round, an agent analyzes contextual data and performance metrics from previous rounds to decide which architectural patterns to (de)activate, with the goal of maximizing overall system performance. This will overcome the combinatorial explosion of possible FL architectural patterns configurations and spare software architects from manual and trial-and-error tuning.

2. Preliminaries

2.1. AP4Fed: A Federated Learning Benchmarking Platform

AP4Fed [1] is a FL benchmarking platform built on top of Flower [6], i.e., a Python library designed to perform FL simulations. AP4Fed extends Flower by integrating architectural patterns and monitoring capabilities to enable performance analysis of FL simulations. The tool uses Docker-Compose to emulate a realistic FL setup, with one container as the central server and multiple containers as clients performing local training and sending updates for aggregation.

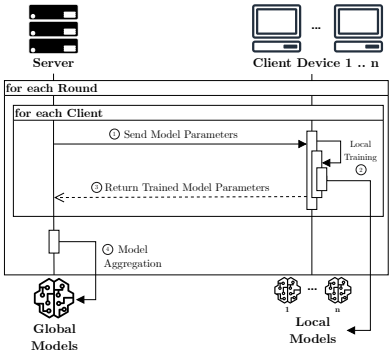

Figure 1: Federated Learning Overview.

	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 comm time
	Total Round Time	Total time per FL round
	Model Accuracy	Aggregated model accuracy

Table 1: Input Parameters and Evaluation Metrics.

FL Simulation. The main steps of a FL 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 FL 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 simulation round to initialize the system's parameters. The *Input Parameters* include NUM_ROUNDS (number of FL 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.

Architectural Patterns. Currently, AP4Fed supports the following architectural patterns [2]: the *Client Registry* pattern provides a centralized storage of relevant information about clients; the *Client Selector* pattern filters participating clients at each round based on resource or data distribution type; the *Client Cluster* pattern groups clients by shared characteristics; the *Message Compressor* pattern compresses data exchanged between the clients and the server.

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

2.2. AI Agent 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 [5]. 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 [5].

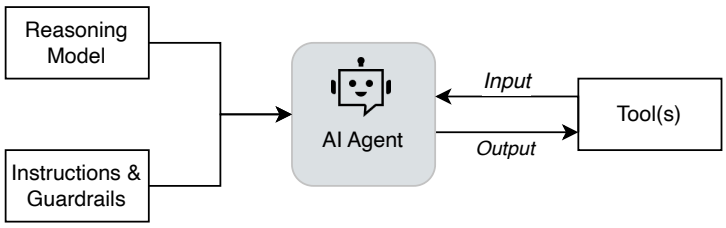

Figure 2: Overview of a Single-AI-Agent main Components (adapted from [5]).

Engineering the AI Agent. Figure 2 depicts an overview of a Single-AI-Agent architecture [5]. 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 [5]. It ingests an input (e.g., contextual data) and produces an output (e.g., a reasoned plan); (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. *Tool(s)* are aimed to interact with the AI Agent, thus acquiring data as input, reasoning over it, and then issuing commands as output.

AI Agent in Federated Learning. Let us consider the goal of pursuing the *performance optimization* in FL systems. A traditional static configuration of architectural patterns acts like a fixed checklist, applying the same setup each FL 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 ability 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 FL system performance that represents our goal.

3. Conceptualization

The approach extends AP4Fed with an AI Agent that monitors and reasons on real-time system performance and contextual information, to dynamically (de)select architectural patterns during FL simulations. This implies both gains and pains that we discuss in the following.

3.1. Engineering the AI Agent for Architectural Pattern Selection

Figure 3 depicts an overview of the proposed approach. The AI Agent is based on a Single-AI-Agent architecture [5], which includes the three core components described in Section 2.2. The *Reasoning Model* is in charge of interpreting the *Input Parameters* and *Evaluation Metrics* of the FL 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 [4]). 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

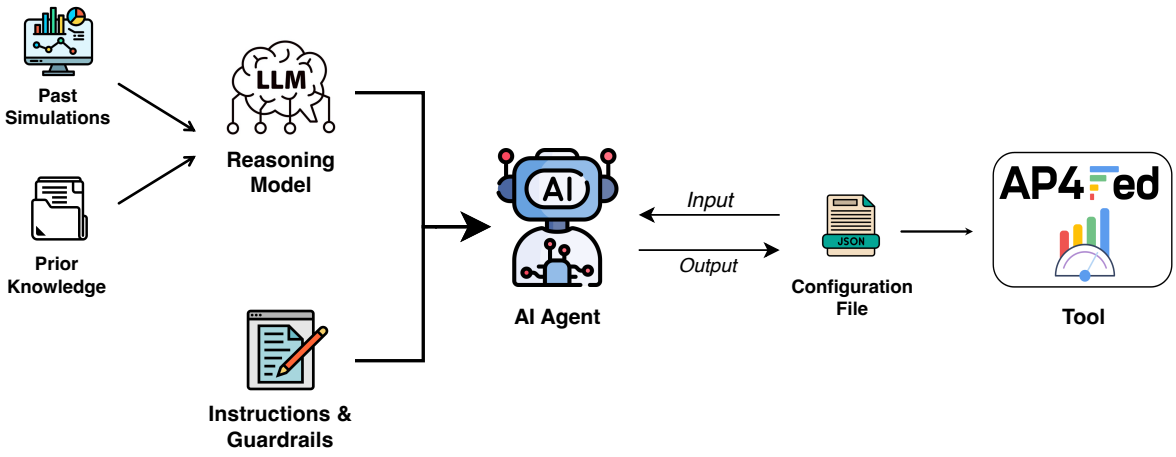

Figure 3: Overview of the Proposed Approach.

prompt that defines the goal to achieve. For instance, we are interested in a selection of architectural patterns that contributes to the FL system performance optimization. A preliminary example of the prompt is provided at the following link: https://hackmd.io/@IvanComp/AIAgentPrompt.

3.2. Expected Results and Open Challenges

Our approach paves the way for the automated selection of architectural patterns while reasoning on the current system parameters and aiming to optimize the performance of FL 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 FL system performance. However, the introduction of an AI Agent 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 [5]. 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 [5]. 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 of the AI agent itself may not be negligible, hence minimizing its impact is auspicable [5].

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] L. Baresi, G. Quattrocchi, N. Rasi, Open Challenges in Federated Machine Learning, IEEE Internet Comput. 27 (2023) 20–27.

[4] 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.

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

[6] 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).