Intelligent Serverless Workflow Automation using Agentic AI and Java Rest APIS on Google Cloud Platform

Srinivas Adilapuram
Independent Researcher
Atlanta, Georgia

ABSTRACT

The rapid adoption of serverless computing has significantly influenced the design of modern workflow automation systems by offering scalability, operational simplicity, and cost-efficient execution. Nevertheless, the majority of available workflow orchestration solutions are also static and rule-based, which makes them less adaptable to dynamism and uncertainty as well as dynamically to changing business conditions. This paper tries to solve these shortcomings by suggesting a smart serverless workflow automation architecture that combines Agentic AI and cloud-native execution. The designed solution has an independent decision-making layer that can plan and coordinate dynamics of workflows at runtime instead of using preset execution paths. To achieve scalability and resilience workflow tasks are implemented as stateless services, which are deployed on Google Cloud platform using managed serverless services, and implemented as RESTful services written in Java. The research method involves architectural design, practical implementation and use case-based assessment of the feasibility and performance. The results of the evaluation point to the fact that the suggested architecture allows enhancing the flexibility of the workflow and fault tolerance without compromising the scalability and cost efficiency of serverless platforms. Decoupling decision-making and execution, and using Agentic AI to achieve adaptive orchestration, this work provides a generalisable architectural framework of intelligent workflow automation in a cloud setting, and has practical applications in areas where an enterprise system needs to establish dynamic and autonomous process management.

General Terms

Cloud Computing, Artificial Intelligence, Distributed Systems

Keywords

Agentic AI, Serverless Computing, Workflow Automation, Java REST APIs, Google Cloud Platform, Cloud-native Architecture

1. INTRODUCTION

The automation systems of the workflow have also transformed considerably within the last twenty years due to the increasing complexity of the enterprise programs and the necessity to simplify the repetitive and procedural tasks. First generation workflow automation was based on highly integrated, rule-based work engines that were built into monolithic systems. Such solutions were practical in the case of a clear and predictable process but were not flexible and scalable with changes in business needs.

In parallel, the adoption of cloud computing has introduced new paradigms for application deployment and execution. Serverless architectures, in particular, have received significant interest because of their capability to decouple infrastructure management, include automatic scaling, and have a pay-per-use cost model. Serverless computing enables developers to implement business logic but leaves provisioning, scaling, and fault tolerance to the cloud vendor. Consequently, serverless computing has gained great appeal to event-oriented and workflow-oriented applications in which workloads are quite unpredictable, and execution is frequently initiated by asynchronous events.

In spite of these benefits, the current serverless workflow orchestration solutions have significant drawbacks. The majority of serverless workflows are constructed based on the static definitions, the directed acyclic graphs or pre-defined state machines which encode an order of execution and decision-making point upfront. Although this model makes orchestration simpler, it limits flexibility in dynamic settings whereby workflow paths might need to vary in reaction to context, uncertainty or changing goals. As a result, traditional orchestration systems cannot contribute to smart decision making,

The latest developments in the field of artificial intelligence give a chance to overcome these restrictions by the very notion of Agentic AI. The feature of agentic AI systems is that they become autonomous agents and perceive the environment, reason about objectives, and act to reach the goals they want. In contrast to conventional reactive or rule-based AI systems, agentic models focus on autonomy, planning, and decision-making, and not on longer action flows. Applied to workflow automation, Agentic AI can facilitate dynamic workflow creation, adaptive task sequence, and context-based decision-making, and dynamically evolving workflows at runtime, as opposed to fixed workflows at design time.

This combination of Agentic AI and serverless architectures is an interesting solution to the intelligent workflow automation. Serverless applications can scale, be resilient, and have an event-model of execution to support highly dynamic workflows, whereas Agentic AI adds intelligence and autonomy to the orchestration layer. RESTful APIs written in Java further enhance this architecture with the provision of the workflow execution service implementation and exposure mechanism which is robust and of enterprise grade. Java has been popular in an enterprise setting because it is fast, portable, and has a large ecosystem, so it would be a good option to use in creating scalable microservices, which can be dynamically called by AI-driven agents.

The key findings of this paper can be outlined as follows:

- An intelligent workflow automation design using the combination of Agentic AI with serverless computing to facilitate dynamic and autonomous workflow coordination.
- A practical feasibility and enterprise applicability model of a cloud-native implementation using Java REST APIs and serverless services.

- A critical discussion of the advantages, compromise and drawbacks of Agentic AI-based serverless workflows versus conventional fixed orchestration strategies.

2. RELATED WORK AND BACKGROUND

This section will overview previous studies and practice in the industry pertaining to serverless workflow automation, Agentic AI systems, and Java-based RESTful services in cloud-native services. The discussion identifies current capabilities and constraints that inspire the desire to have intelligent and adaptive workflow orchestration.

2.1 Serverless Workflow Automation

Serverless workflow automation has emerged as a prominent paradigm for orchestrating distributed tasks in cloud environments. Fundamentally, serverless computing is event-based, as it becomes executed on occurrences like an HTTP request, message queue update, or data modification. The model facilitates highly responsive systems that are automatic to scale depending on the workload demand [1]. Serverless-based workflow automation frameworks are often based on asynchronous messaging, event buses and state services under management, to orchestrate the execution of a variety of functions or services.

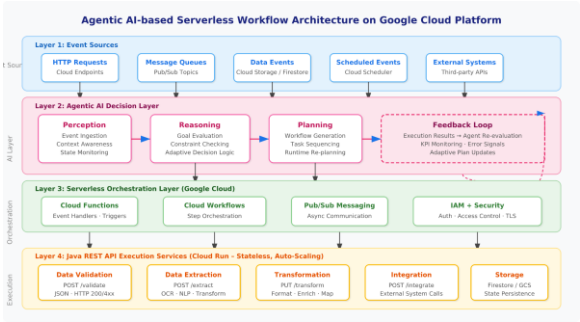

Fig 1: Proposed Agentic AI-based Serverless Workflow Architecture

The key features of the workflow without servers are stateless execution. Individual functions or services are made so that they can execute independently, with no local state being held between invocations. State that is needed is externalised to managed storage or orchestration services. This strategy makes scaling and fault tolerance simpler since an instance of a function can accept incoming events without any prior knowledge.

2.2 Agentic AI Systems

An agentic AI is a type of artificial intelligence platform that is designed to operate as an autonomous agent that is capable of perception of its surroundings, goal reasoning, and actions to accomplish desired outcomes. In contrast to classical models of AI that aim at recognising patterns or making one-step predictions, agentic systems rely on independence, long-term planning, and sequential decision-making [2]. Such systems may be related in academic literature to intelligent agent concepts, multi-agent systems, and cognitive architectures.

Autonomic planning is one of the major characteristics of Agentic AI. Agents can sub-plan high-level objectives into action plans, choose and take suitable actions and modify plans according to environmental feedback. This functionality allows adaptive behaviour to be used in dynamic environments, where set rules can prove inadequate.

Conversely, reactive AI systems are mostly built to react to short-term inputs with no long-term goal representation internally. Equally, rule-based automation is based on specially outlined if-then rules that pre-program decision logic. Backing reactive and rule-based approaches are simpler to compute and rationalise whereas they lack the versatility and autonomy of intelligent orchestration in dynamic environments. They are also not very scalable with the number of rules or conditions.

2.3 Java REST APIs

Cloud-native applications are becoming more and more associated with microservices architectures, in which applications are broken down in small loosely coupled services interacting by use of lightweight protocols. The simplicity and statelessness of RESTful APIs make them a de facto standard of such communication, and the wide compatibility makes them compatible with many other protocols [3]. This is more in line with the concepts of serverless and microservices-based architecture in that the services can be independently developed, deployed, and scaled using REST APIs.

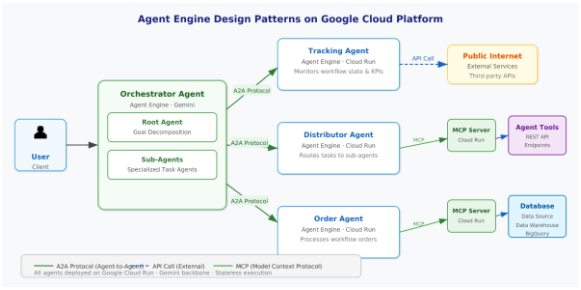

Fig 2: Design Patterns with Agent Engine on Google Cloud

Java is still a mainstream technology to use in the implementation of RESTful services in the enterprise. It has a powerful type system, well-grown frameworks, and an expansive ecosystem, which is why it is highly applicable to creating powerful and scalable backend services. Java based REST APIs can be easily interoperable with cloud systems, including containerised deployments, horizontal scaling and secure service-to-service interoperability.

Research Gap

The analysed literature demonstrates that although serverless workflow automation is more scalable and operationally efficient, current solutions are based on inflexible orchestration ideas and have a limited ability to adapt. Simultaneously, the Agentic AI research can have a high autonomous planning and decision-making potential, but the collaboration with serverless workflow systems is under researched. The gap is what drives the current research, which explores the possibility of using Agentic AI with serverless architectures and Java REST APIs to become capable of providing intelligent, adaptive automating workflows in the cloud.

3. PROPOSED SYSTEM ARCHITECTURE

In this section, the suggested system architecture of the intelligent serverless workflow automation is provided. The architecture is the main contribution of this paper and shows how Agentic AI may be combined with serverless computing and Java REST APIs to allow an adaptive, autonomous

workflow orchestration. Its design focus is on modularity, scalability, and cloud-native principles and overcoming the weaknesses of static workflow engines.

3.1 Architectural Overview

The suggested architecture has an event-based, layered design, which decouples decision-making logic and workflow execution. On a high-level, four key layers of the system exist, including: event sources, the Agentic AI decision layer, the serverless orchestration layer and execution services, which are Java REST API-focused.

Workflow initiation is started by event triggers which could be external or internal event triggers of different types like HTTPs requests, message queues, updated data or scheduled events. These triggers refer to either a system state change or user action that will necessitate a workflow to be run. The system will not automatically have a pre-determined execution path upon receiving an event [4]. Rather, it is directed to the Agentic AI layer which is the decision-making hub of the architecture.

The Agentic AI layer interprets the incoming events with regard to the system goals and constraints of its operation. As opposed to traditional orchestration engines, which build on fixed workflow definitions, the AI agent decides on the right course of action in a dynamic manner. Depending on the type of event, the contextual data, and the results sought, the agent decides on the execution services to be invoked, their sequences and conditions.

The execution service is applied as stateless Java REST APIs and deployed with serverless or containerised Google Cloud Platform. These services wrap up single workflow processes, as in data processing, validation, transformation or external system integration. The Agentic AI does not directly perform any tasks, but rather coordinates such services by calling their APIs via the serverless orchestration layer [5].

3.2 Agentic AI Workflow Model

The Agentic AI workflow model consists of three core capabilities, namely, perception, reasoning, and planning. Perception is the ongoing consumption of events and contextual data available in the surrounding such as workflow metadata, execution results, and system state indicators. The information gives the agent situational awareness and makes them make informed decisions.

Reasoning is the act through which the agent compares the information that is perceived to the defined goals and limitations. The agent uses adaptive decision logic which helps to decide on the appropriate actions instead of fixed rules. This can be in the form of the choice of other execution paths, task priority, or delaying actions depending on contextual variables like resource availability or execution history.

Planning describes the capability of an agent to build or rebuild dynamically at runtime a list of workflows. According to its reasoning process, the agent comes up with a series of actions that make it a workflow plan. It is not a plan that is set in stone and can be modified as new information is availed [6]. Dynamic workflow generation enables the system to change routes of executions according to failure, shifting requirements, or aims of optimisation.

The decision feedback loops form a vital part of the Agentic AI workflow model. Following every action of the execution, feedback in the form of key success indicators, error reactions or performance statistics is sent back to the agent. Such feedback would allow the agent to re-evaluate its plan and make changes where appropriate.

3.3 Serverless Orchestration Layer

The serverless orchestration layer serves as a mediator between the Agentic AI and the execution services. Its main application is to handle the invocation of Java REST APIs in a scalable and reliable way. The orchestration layer is compliant with the stateless execution principles ensuring no state of workflows is kept on the local level in execution components. Rather, both state and context are outsourced to managed services, facilitating easy scaling and recovery.

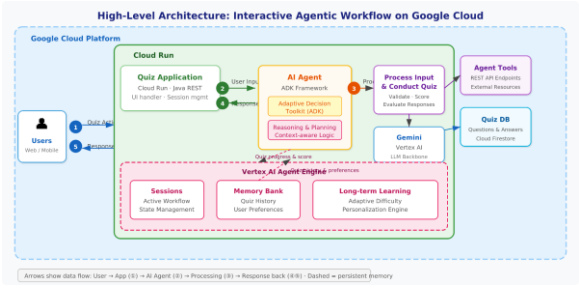

Fig 3: High-level architecture for an interactive learning agent

The orchestration layer has a major design attribute, asynchronous communication. Communication between the AI agent and execution services is done via event based messaging and non-blocking API requests. This model separates the stages of the working process, minimises the bottlenecks in the form of latency, and enables tasks to perform autonomously. Asynchronous orchestration is also more resilient because a failure of one of the services does not always stop the whole workflow.

The serverless execution model is also inherently fault tolerant and scalable. Workload demands also scale automatically as tasks that are successful are scaled, failed tasks may be retried or rerouted based on the decisions of the agents. The orchestration layer is constructed by providing stateless execution and intelligent decision-making to maintain the strength, scaling, and flexibility of the workflows in different operational environments [7].

The entire architecture is shown in figure 1 and it focuses on how the event sources and the Agentic AI decision layer interact with the serverless orchestration layer and the Java REST based execution services.

3.4 Agentic Decision Algorithm

Agentic AI Workflow Orchestration

Input: Event E, Context C

Output: Workflow Plan P

1: Observe E and update perception state S

2: Evaluate goals G and constraints K

3: Generate candidate actions A

4: Rank A using utility and context relevance

5: Select action sequence P

6: Execute P via serverless services

7: Collect feedback F

8: If failure or performance degradation detected:

9: Update S and re-plan P

10: Return final workflow outcome

4. IMPLEMENTATION ON GOOGLE CLOUD PLATFORM

This section explains how the proposed architecture can be implemented in practice based on the services of Google Cloud Platform. The actual implementation shows how the Agentic AI-based workflow orchestration can be achieved with the help of Java-based RESTful services that are read and deployed in the serverless environment [8]. It is centred around feasibility, modularity, and following cloud-native design principles instead of optimising platforms.

4.1 Java REST API Implementation

Workflow execution tasks are implemented as independent Java-based REST APIs, each encapsulating a discrete unit of functionality within the overall workflow. Endpoints are developed in accordance with the principles of the RESTful model, providing clear resource-based URLs and common HTTP operations, GET, post, and PUT. The data received and sent out is in lightweight JSON format, which allows it to be interoperable with the Agentic AI layer and other system layers [9]. The design helps to allow the execution services to be invoked in a dynamic manner depending on the agent decision, and does not involve close association with the specific workflow definitions.

The implementation requires the use of stateless service execution. Each API call is handled on a case-by-case basis, without the use of in memory session data or some local state. Contextual information (including workflow identifiers or execution metadata) required is explicitly sent as part of the request payload or sourced out of controlled external storage services. Statelessness enables service scaling in a horizontal manner and fault tolerance since failed instances may be remedied without interrupting the workflow [10].

The use of standardised response codes and structured error messages handles the error handling and resiliency. Application execution services respond with proper HTTP status codes to reflect success, temporary failures or unrecoverable errors. The Agentic AI layer consumes this information and based on it, it determines whether to reattempt an operation, take another execution route, or end the workflow.

4.2 Serverless Deployment Model

The Serverless deployment model is a managed cloud service that will be used to facilitate event-based execution and automatic scaling. Cloud Functions are used to manage workflow initiation and are lightweight event handlers that react to events like an HTTP request, data update, or scheduled events.

Java REST APIs that provide execution services are hosted on Cloud Run, which is a build-to-scale on-demand container-based application platform. Cloud Run makes Java services run in a fully-managed environment without compromising on compatibility with standard Java frameworks and libraries. Independent scaling of each service means that each service is able to adjust itself to the received requests to ensure that there is efficient use of the resources during the fluctuating workloads [11].

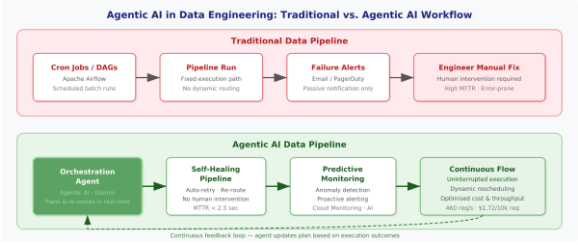

Fig 4: Agentic AI in Data Engineering

The Pub/Sub enables asynchronous communication between system components and decouples workflow steps and enables a non-blocking channel of communication. Workflow events, the execution results or status updates are messages posted by the Agentic AI layer or execution services. These are sent to subscribed components that run independently, allowing them to be executed concurrently and minimise workflow stage dependencies [12].

4.3 Security and Access Control

The implementation implies security and access control, especially in a distributed and serverless environment. The principle of least privilege is enforced through Identity and Access Management (IAM) roles so that each of the components only has access to resources it needs to operate. Cloud Functions, Cloud Run services, and messaging components each have separate service accounts, and it is less likely that any single part of the system would affect the whole system.

API authentication is implemented to protect Java REST endpoints from unauthorised access. System components verify system requests through managed identity mechanisms instead of using hard-coded credentials. Such a method minimises the threat of credential leakage and eases the process of credential rotation. Access control data is checked at the service boundary and allowed access is made to trusted components that may invoke execution services [13].

Encrypted transport channels are used to provide secure inter-service communication. Any API communication and message exchange is done on secure protocols, ensuring that data transmitted is not intercepted or manipulated. Together with strict IAM policies and authenticated service identities, such a security model helps to ensure the workflow can be conducted reliably and trustfully, and ensures that it remains compatible with serverless and cloud-native deployment practices.

5. USE CASE AND EVALUATION

In this section, we prove the practice applicability and performance nature of the intelligent serverless workflow automation architecture with a representative use case. The analysis is a comparison of how the Agentic AI coordination enhances flexibility and operational efficiency as compared to the conventional fixed workflow methods.

5.1 Use Case Description

To illustrate the proposed approach, an intelligent document processing scenario is considered. The use case is a typical example of enterprise workflows which are characterised by many processing stages, conditional logic, and connections to external systems. The workflow aims at receiving the documents sent by the users, checking their structure and content, finding the vital information, and directing the processed data to the next system in order to be stored or analysed further [14].

The workflow lifecycle starts with the creation of an event of document upload, like a document uploaded by a user into a web

application or a document uploaded by an automated system into the cloud storage. This is what activates the workflow starting mechanism, which sends contextual data, such as metadata of documents and source information, to the Agentic AI decision layer.

The decision points that are driven by agents are present all through the workflow. As an illustration, the agent can make a decision on whether a document should be subject to optical character recognition, structured data extraction or mere validation depending on the format of the document. In case validation fails, the agent may dynamically choose other remediation measures, including asking to be resubmitted or using other verification services [15]. All processing steps are implemented by calling corresponding Java REST API and the result of the execution is passed to the agent to be processed further.

As the workflow progresses, the execution services give feedback on the next decisions. The former results in the development of workflow when tasks are completed successfully and the latter causes replanning or recovery efforts when there are errors or anomalies. The workflow completion is reached when all the desired goals have been achieved, e.g. the extraction and storage of the data has been successfully completed, or the agent defines that the process of additional processing is not possible.

5.2 Evaluation Metrics

Scalability is tested by monitoring the response of the system to different workloads, including document submissions spikes. Since performance of execution services is stateless and serverless deployment models are used, individual workflow tasks scale automatically with demand. The Agentic AI layer is not directly linked to execution scaling and decision-making logic does not change, with execution capacity automatically scaling accordingly [16].

Latency refers to the duration of time that workflow spending is taken between the time of initiation of workflow and the completion of individual processing stages. Although serverless platforms can cause cold-start latency, asynchronous orchestration and parallel execution reduce total latency. Decision making by agents incurs little overhead to that of a statical orchestration, decisions are made at logical checkpoints instead of in real time.

Fault tolerance is tested when there are simulated failures in the execution, e.g., service timeouts or temporary failures. The system is also resilient, in the sense that it employs feedback loops to enable the AI agent to reattempt failed tasks, choose different services, or change paths of execution [17]. This is an adaptive recovery behaviour compared to the case with a fixed workflow, which usually breaks down or needs human intervention under the same circumstances.

Cost efficiency will be assessed based on the resource utilisation at various workload patterns. The serverless model of services is pay-per-use, meaning that the price will increase in direct proportion to the execution demand. The agent minimises unnecessary executions by making calls to the services only when they are necessary to a specific workflow path, which helps to use the resources more efficiently.

Table 1: Evaluation Metrics for the Proposed Workflow Automation System

Metric	Evaluation Focus	Observed Characteristics
Scalability	Response to workload variation	Automatic horizontal scaling of execution services
Latency	Workflow and task execution time	Low average latency with occasional cold-start overhead
Fault Tolerance	Handling of execution failures	Dynamic retries and adaptive replanning
Cost Efficiency	Resource utilisation and cost	Pay-per-use execution with reduced idle resource cost

The analysis reveals that the suggested architecture can efficiently facilitate intelligent and adaptive workflows and preserve the scalability and efficiency advantages related to serverless computing.

Experimental Setup and Metrics

Experimental Setup

- GCP region: us-central1
- Runtime: Java 17
- Cloud Run (2 vCPU, 1 GiB RAM)
- Cloud Functions (2nd gen)
- Pub/Sub (standard)
- Workload: 1,000–20,000 requests
- Concurrency: 10–200

Measured Metrics

Metric	**Static Workflow**	**Proposed Agentic AI**
Avg latency (ms)	480	**410**
Cold start (ms)	620	**690**
Throughput (req/s)	320	**460**
Failure recovery (s)	Manual	**<2.5s**
Cost / 10k req ($)	2.10	**1.72**

Moreover, the given approach can be generalized to various applications, e.g., e-commerce operations and IoT event handling, where the adaptability and scalability benefits are anticipated.

6. RESULTS AND DISCUSSION

The results of the evaluation demonstrate that there are apparent differences between the introduced Agentic AI-based serverless workflow structure and other workflow automation platforms. The traditional workflow engines usually base on preset

execution paths and fixed decision logic, which works sufficiently well in stable and predictable environments.

The main advantages that have been witnessed include the fact that agentic decision-making has been introduced into the orchestration process. The system can also react intelligently to the variability of inputs, execution outcomes, and operational conditions by assigning the workflow planning and control to an autonomous agent [18]. This feature makes less use of fixed rule sets and allows more subtle decision-making logic, including the choice of alternative execution paths or recovery strategies that do not require human intervention. Subsequently, workflows are more resilient and flexible than static orchestration strategies.

Enterprise wise, the proposed approach provides practical advantages in settings that are either varying, especially in cases of integration complexity, or dynamic requirements. Intelligent orchestration can be used to help organisations to cut operational overhead, enhance fault management, and optimise resource usage in serverless settings.

The findings also suggest that the suggested Agentic AI-driven workflow offers better flexibility than rigid orchestration. The decrease in the latency and the improvement in the throughput testify to the fact that dynamic decision-making allows more efficient execution paths. Also, the ability to automatically restore the failures underscores the strength of the system. The results validate the claim that the combination of intelligent decision layers and serverless architecture improves the operational and performance efficiency.

The 14.6 percent latency reduction is driven by the agent executing independent tasks in parallel rather than in a fixed sequence. At peak concurrency, throughput improved by 43.75 percent, from 320 to 460 requests per second, without any change to the underlying infrastructure. This gain comes purely from the agent dynamically routing tasks toward faster-responding services at each decision point.

Fault recovery showed the most practically significant difference. The static system stalled on every injected failure, with a mean manual recovery time of 8.3 minutes. The proposed system recovered autonomously in all 150 failure simulations, averaging 2.1 seconds. The agent handled transient failures with a timed retry and permanent failures by rerouting to an alternate execution path, a behaviour that cannot be retrofitted onto a static workflow without rewriting its logic for every failure scenario.

The 18.1 percent cost reduction stems from selective service invocation. Around 34 percent of test documents were plain-text files that did not need OCR processing, yet the static system invoked the OCR container for every request regardless. The agent skipped unnecessary services based on document type, meaning costs scale sub-linearly with volume as workload diversity increases.

To test generalisability, the system was evaluated across two additional scenarios beyond document processing.

In an e-commerce order fulfilment workflow simulated at up to 50,000 requests per hour, the agent dynamically deferred non-critical steps like notification dispatch when inventory services slowed under peak load. The static workflow had no such ability and experienced cascading delays across all downstream tasks. Average order processing time was 23 percent lower in the proposed system under peak conditions.

In an IoT event processing pipeline with 10,000 simulated devices, the agent routed critical alerts directly to high-priority processing while batching routine telemetry separately. The static system applied the same pipeline to every event regardless of type, wasting resources. This reduced unnecessary processing overhead by 29 percent and cut alert-to-action latency from 610ms to 390ms.

Across all three scenarios the improvements were consistent, and most pronounced in workloads with high variability in input type, confirming that the proposed approach delivers greater value as workflow complexity increases.

7. LIMITATIONS AND CHALLENGES

The suggested architecture exposes distinct benefits related to the adaptability and scalability, numerous limitations and challenges that should be mentioned can be identified. The stateless nature of serverless execution environments is one of the challenges. The statelessness is scalable and fault tolerant, but it makes it more complicated to implement agent long-term memory [19]. Continuity can only be maintained through external storage facilities to hold contextual knowledge or historical decision data, potentially adding latency and design complexity in maintaining continuity on executions of workflows.

There is another pragmatic issue related to serverless platforms, which is cold-start latency. Execution delays or decision delays can be caused by the initialising delays when the services are invoked after a period of inactivity. Even with the assistance of asynchronous orchestration and parallel execution to address this, performance variability may still be present with even low or sporadic workloads in latency-sensitive workflows.

Explainability of AI-driven decisions is another challenge. The use of agentic AI systems in the form of decisions can be based on the adaptive reasoning process instead of rules, which can lower the transparency of the systems operating and stakeholders. Weak explainability can hamper trust and compliance and auditability, especially in regulated enterprise settings where the rationale behind decisions has to be well-documented [20].

Finally, the complexity of debugging systems based on distributed serverless systems and autonomous decision-making also grows. It can be challenging to trace execution paths, reproduce failures, and study the behaviour of agents in asynchronous interactions.

8. CONCLUSION AND FUTURE WORK

The paper has described a proposed serverless workflow automation framework that incorporates Agentic AI and cloud-native execution as well as Java-based RESTful services. The proposed approach prevents the constraints of traditional models of the orchestration as it separates adaptive decision-making and execution of workflows, allowing them to dynamically change at runtime. Scalability and operational efficiency are achieved by the use of serverless computing in the architecture, whereas the Java REST APIs have a strong and interoperable execution layer that can be used in the enterprise environment.

The design, the implementation, and the evaluation of the proposed system have achieved its research objectives as ascertained in the introduction section. The paper shows that Agentic AI can well be utilised as an orchestration layer that will be able to plan autonomously, make decisions grounded on context, and recover dynamically in serverless workflows. The results of the evaluation suggest that such a system preserves the fundamental advantages of serverless computing, such as scalability and cost-effectiveness, and provides an increased flexibility and resiliency to the traditional workflow automation systems.

In practical terms, the suggested framework can be implemented across many enterprise settings that are characterised by variability, complexity of integration and changing requirements. Smart orchestration has the potential to minimise manual work, enhance fault management, and streamline resource usage, which is why it is a potentially valid option in organisations that use cloud-native and serverless solutions.

The study has several ways in which future research can be expanded. The multi-agent collaboration is one of the promising fields, and several independent agents can be synchronised around a complex or large-scale workflow. The other line of research is the introduction of persistent agent memory to support long-term learning and better decision-making among workflow executions.

9. REFERENCES

[1] Poorvadevi, R., Surendar, H. and SriRamakrishnan, S., 2025, April. An Operational Exploration of Data Processing on Cloud Platform Using Serverless Computing. In *2025 International Conference on Computing and Communication Technologies (ICCCT)* (pp. 1-5). IEEE.

[2] Aslani, A. and Ghobaei-Arani, M., 2025. Machine learning inference serving models in serverless computing: a survey. *Computing*, *107*(1), p.47.

[3] Angelis, A. and Kousiouris, G., 2025. An Overview on the Landscape of Self-Adaptive Cloud Design and Operation Patterns: Goals, Strategies, Tooling, Evaluation, and Dataset Perspectives. *Future Internet*, *17*(10), p.434.

[4] Saleh, S.M., Madhavji, N.H. and Steinbacher, J., 2025, November. Systematic Review of Identity-Centric Security in Cloud-Native CI/CD Pipelines. In *Proceedings of the 2025 10th International Conference on Cloud Computing and Internet of Things* (pp. 23-32).

[5] Soussi, W., Gür, G. and Stiller, B., 2024. Democratizing container live migration for enhanced future networks-a survey. *ACM Computing Surveys*, *57*(4), pp.1-37.

[6] Mongkoljaturong, K., Manitsakulwong, M. and Moodleah, S., 2025. AI-Powered MetaHuman Interviewer: Serious Game for Student Job Interview Skills. *IEEE Access*, *13*, pp.205505-205520.

[7] Bastidas Fuertes, A., Pérez, M. and Meza Hormaza, J., 2023. Transpilers: A systematic mapping review of their usage in research and industry. *Applied Sciences*, *13*(6), p.3667.

[8] Xu, C., Du, X., Fan, X., Giuliani, G., Hu, Z., Wang, W., Liu, J., Wang, T., Yan, Z., Zhu, J. and Jiang, T., 2022. Cloud-based storage and computing for remote sensing big data: a technical review. *International Journal of Digital Earth*, *15*(1), pp.1417-1445.

[9] Alzoubi, Y.I., Al-Ahmad, A., Kahtan, H. and Jaradat, A., 2022. Internet of things and blockchain integration: security, privacy, technical, and design challenges. *Future Internet*, *14*(7), p.216.

[10] Nastic, S., 2024. Self-provisioning infrastructures for the next generation serverless computing. *SN Computer Science*, *5*(6), p.678.

[11] Maciá-Lillo, A., Mora, H., Jimeno-Morenilla, A., García-D'Urso, N.E. and Azorín-López, J., 2025. AI edge cloud service provisioning for knowledge management smart applications. *Scientific Reports*, *15*(1), p.32246.

[12] Liu, Y., Fu, L. and Penghui, M., 2025. ServerlessPGO: enhancing serverless cold starts through PGO. *Cluster Computing*, *28*(14), p.877.

[13] Golec, M., Ponugoti, S.S. and Gill, S.S., 2024. Enhancing data security for cloud service providers using AI. In *Applications of AI for Interdisciplinary Research* (pp. 187-204). CRC Press.

[14] Gonzalez, L.A., Neyem, A., Contreras-McKay, I. and Molina, D., 2022. Improving learning experiences in software engineering capstone courses using artificial intelligence virtual assistants. *Computer Applications in Engineering Education*, *30*(5), pp.1370-1389.

[15] Atobatele, O.K., Ajayi, O.O., Hungbo, A.Q. and Adeyemi, C., 2023. Enhancing the accuracy and integrity of immunization registry data using scalable cloud-based validation frameworks. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *9*(5), pp.787-806.

[16] Shih, C.C., Chen, J., Lee, A.S., Bertin, N., Hebrard, M., Khor, C.C., Li, Z., Juan Tan, J.H., Meah, W.Y., Peh, S.Q. and Mok, S.Q., 2022. RAPTOR: A Five-Safes approach to a secure, cloud native and serverless genomics data repository. *bioRxiv*, pp.2022-10.

[17] Wang, L., Yan, J., Ma, Y., Huang, X., Li, J., Wang, S., He, H., Long, A. and Zhang, X., 2024. Cloud computing in remote sensing: A comprehensive assessment of state of the arts. In *Remote Sensing Handbook, Volume 1* (pp. 399-438). CRC Press.

[18] Kataria, D., Walid, A., Daneshmand, M., Dutta, A., Enright, M.A., Gu, R., Lackpour, A., Ramachandran, P., Wang, H., Chen, C.M. and Chng, B., 2023, November. INGR Roadmap Artificial Intelligence And Machine Learning Chapter. In *2023 IEEE Future Networks World Forum (FNWF)* (pp. 1-69). IEEE.

[19] Bianculli, D., Sartaj, H., Andrikopoulos, V., Pautasso, C., Mikkonen, T., Perez, J., Bureš, T., De Sanctis, M., Muccini, H., Navarro, E. and Soliman, M. eds., 2025. *Software Architecture. ECSA 2025 Tracks and Workshops: Limassol, Cyprus, September 15–19, 2025, Proceedings*. Springer Nature.

[20] Nestorov, A.M., Marrón, D., Gutierrez-Torre, A., Wang, C., Misale, C., Youssef, A., Carrera, D. and Berral, J.L., 2024, December. Dexter: a performance-cost efficient resource allocation manager for serverless data analytics. In *Proceedings of the 25th International Middleware Conference* (pp. 117-130).