



A Reactive Serverless Fabric for Autonomous Workflow Tuning

1st Bharath Somu , 2nd Hiroshi Tanaka

Bharath Somu, Senior Engineer ,bharthsomu@gmail.com

Hiroshi Tanaka , Research Assistant

Abstract—Workflow-oriented applications in multi-cloud environments increasingly face variable demand, elastic scaling needs, and structural diversity, making serverless models attractive for their low runtime costs and ease of use. Autonomous workflow execution, however, remains a persistent challenge: prior survey work shows that managing complex, heterogeneous workflows across clouds is daunting, and the steep learning curve often discourages adoption. Event-driven approaches address this by letting users focus on descriptive specifications rather than procedural implementation details. Yet despite the growing popularity of event-driven cloud offerings, recent survey findings point to a lack of system support for complex workflows that inherently depend on multiple triggers. The high execution frequency typical of serverless functions further intensifies the need for robust event-driven support. Recognizing both the scale and distributed nature of these functions, we design a serverless architecture capable of managing complex agent-based workflows economically and automatically, in which every action module is realized as a self-contained, event-triggered function operating in a serverless manner.

We propose the Event-Driven Serverless Orchestration (EDSO) framework, which autonomously manages a complex distributed agent application structured as a closed-loop, event-driven control system. Autonomy is achieved through self-adaptation mechanisms that combine real-time reactive scheduling with production-distribution optimization. EDSO functions primarily as a central orchestrator coordinating other functions responsible for executing major agents. The orchestrator itself handles three core modules: an outer-agent trigger that responds to cloud environment notifications, a real-time scheduler activated upon completion of triggered agents within the outer agent's cloud zone, and a major-agent-control optimization function that ensures resource-efficient operation of the major agents—generating execution events for the inner-agent set and managing their distribution.

Keywords—Event-Driven Computing, Serverless Orchestration, Cloud Workflow Automation, Autonomous Cloud Operations, Function-as-a-Service (FaaS), Workflow Optimization, Cloud-Native Architecture, Event Streaming and Messaging, Intelligent Resource Scheduling, Distributed Serverless Systems.

1. INTRODUCTION

Event-Driven Serverless Orchestration Framework for Autonomous Cloud Workflow Optimization

The primary mission of cloud computing is to abstract the technology infrastructure from higher-level applications and services. Despite this high-level abstraction, lower layers still require engineering effort for proper configuration and optimization for diverse execution environments. Current cloud orchestration solutions require semi-static configurations selected by users, requiring low-level knowledge of the services deployed at lower levels, even when enabling access to higher-level software abstractions

such as microservices or serverless functions. Most cloud orchestration solutions are unaware of the execution status of network applications accessed by microservices or by serverless function workflows. Such execution context information should be an input to orchestration engines, and serverless functions, microservices, or other resources exposed by a lower-level for-pay abstraction should be the outcome of executing the orchestration configuration.

A serverless microservice orchestration engine driver architecture provides a reactive communication pattern supporting workflow execution or schedule triggers caused by events that can occur both inside and outside the cloud environment. Such an architecture enables the development of an event-driven serverless orchestration framework that strives for optimally automatic self-configuration and scheduling autonomously and costlessly. The proposed



engine driver architecture uses the base mechanism of the cloud platform and extends the base functionality with three execution context-specific scheduling and optimization algorithms: Real-Time Reactive Scheduling, Serverless Microservice Optimal Configuration and Scheduling, and Hybrid Orchestration Configuration.

Table 1 — Orchestration Engine Event Categories

Event Category	Trigger Source	Orchestrator Response
Outer-Agent Trigger	Cloud environment notification	Invokes outer-agent handling routine
Real-Time Reactive Scheduling	Agent completion within outer-agent's cloud zone	Schedules next-ready jobs immediately
Major-Agent-Control Optimization	Periodic / threshold check	Optimizes resource use across major agents
Explicit Event	User-defined publish to event bus	Delegates task to subscribed microservice
Serverless Execution Flow Control	Function completion / failure signal	Advances or retries workflow step

Table 2 — Scheduling and Optimization Algorithms

Algorithm	Deadline Type	Primary Objective	Overhead
Task Pool Scheduling	Hard deadline	Allocate resources per event arrival using queueing model	Low
Real-Time Reactive Scheduling	Soft / hard	React to threshold breaches, minimize execution overhead	Low

Algorithm	Deadline Type	Primary Objective	Overhead
Serverless Microservice Optimal Config.	Soft deadline	Tune resource allocation to raise deadline-met confidence	Medium
Hybrid Orchestration Configuration	Mixed	Balance processing overhead vs. workflow execution time	Low–Medium

2. Background and Motivation

Independent cloud workflow scheduling and control engines are mostly limited to batch execution and cannot handle non-urgent real-time events occurring during workflow execution either. Companies currently operating in the cloud impose cost optimization manually or in a predefined manner for foreseeable situations, but cannot afford to execute all service requests autonomously because they are still uncertain about the level of scalability and availability that may be needed in the cloud. The proposed approach implements an event-driven serverless cloud workflow architecture, which autonomously optimizes online execution by switching from a non-critical workload condition to a cost-saving mode whenever resource supply is more than sufficient compared to predicted demand. Autopilot-like cost-saving automation eliminates wasted resource consumption during low-traffic periods, while on-demand cloud resources provide the necessary elasticity for real-time control-oriented workload conditions.

Event-driven paradigms have been widely adopted in many domains, such as software development and integration, data processing, system monitoring, building automation, and others. Chaos engineering includes controlled experiments to discover weaknesses in the ecosystem of microservice architectures. APIs in the fifth generation of mobile networks will invoke functions on a serverless basis. However, while these developments cover a multitude of possible applications, event-driven paradigms in the context of cloud workflow management have received



significantly less attention. Furthermore, especially for cloud service-driven active workflows, managing a real-time traffic picture to trigger event functions for critical operation and control workflows or continuous services has mainly remained a manual task.

Table 3 — Real-Time Workflow Classes

Property	Class 1: External-Event Workflows	Class 2: System- Controlled Workflows
Task arrival	Triggered by external events	Controlled by the system itself
Arrival model	Drawn from a known distribution	Generated on demand
Deadline type	Hard, quantized into calendar slots	Soft, probabilistic
Control mechanism	Task Pool Scheduling (queueing model)	Add servers to raise deadline-met confidence
Key trade-off	Resource allocation vs. arrival variability	Confidence level vs. resource cost

3. Architectural Principles

An event-driven cloud workflow orchestration framework is proposed specifically for optimizing the execution of long-running workflows in clouds. Such workflows often consist of densely connected task structures that yield significant overhead in long-distance task-to-task communications. While serverless function-as-a-service architectures optimize such overhead through automatic scaling, existing workflow orchestration and scheduling technologies are not reactive to changes in execution state or task execution time. The proposed event-driven framework maps autonomous optimizations onto event channels, enabling task-level control of algorithm selection, task scheduling, and workflow adaptation in response to execution state changes.

Workflow execution requires both smoke detectors, which detect problems, and firemen, who react to the problems detected. A fireman acts after a problem is detected,

rather than trying to predict problems before they occur. Such an event-driven system consists of a reactive blueprint for execution and broadcast channels that propagate states and changes. A change in state can automatically trigger a composite function in a separate entity. Two types of event-driven control are considered: explicit events and serverless function execution flow control. High-level orchestration is defined as a selection of scheduling, optimization, and adaptation algorithms, where the scheduling and adaptation algorithms execute during workflow execution, with the orchestration engine acting only when an algorithm is completed.

Mathematical Formulas:

Eq. 1 — Resource Utilization Ratio (queueing model)

$$\rho = \frac{\lambda}{c\mu}$$

where λ is the task arrival rate, c is the number of servers, and μ is the service rate per server.

Eq. 2 — Utilization Metric $U(t)$

$$U(t) = \frac{R_{used}(t)}{R_{total}(t)} \times 100\%$$

Eq. 3 — Reactive Mode Selection

$$\text{Mode}(t) = \text{Cost-Saving} \text{ if } U(t) \leq \theta_{low}$$

$$\text{Mode}(t) = \text{Elastic-Scale} \text{ if } U(t) \geq \theta_{high}$$

$$\text{Mode}(t) = \text{Nominal} \text{ otherwise}$$

The orchestrator switches between cost-saving and elastic-scale modes based on threshold crossings of $U(t)$.

Eq. 4 — Cost-Confidence Objective (soft-deadline workflows)

$$C(n) = \alpha n + \beta (1 - P_{meet}(n))$$

α weights the per-server cost and β weights the penalty for missing the target confidence.

Eq. 5 — Deadline-Met Confidence

$$P_{meet}(n) = 1 - e^{-\kappa n}$$

κ reflects how quickly added resources raise confidence, consistent with the paper's diminishing-returns discussion.



Eq. 6 — Total Execution Overhead

$$O_{total} = O_{base} + \sum_{i=1}^N o_i(t_i)$$

O_{base} is the fixed orchestration overhead; $o_i(t_i)$ is the per-task overhead the lightweight algorithm seeks to minimize.

Eq. 7 — Optimal Server Count

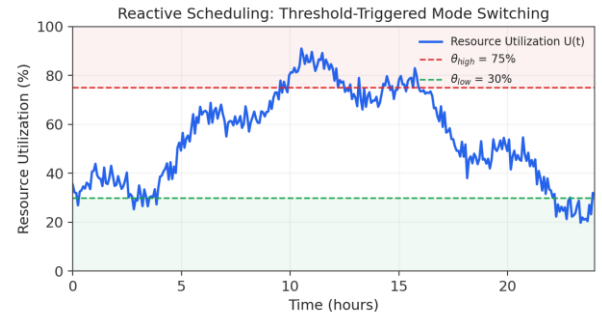
$$n^* = \arg \min_n C(n) \text{ s. t. } P_{meet}(n) \geq P_{target}$$

A. Event-Driven Paradigms

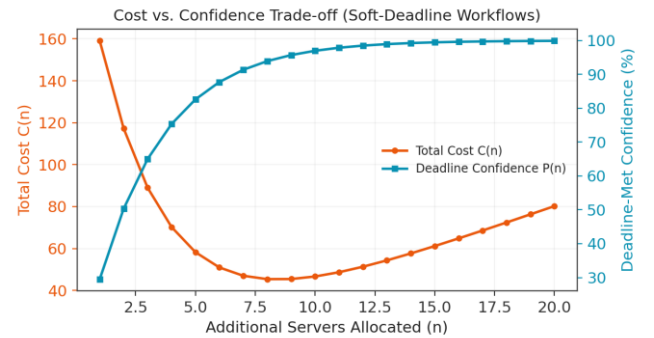
Achieving intelligent automation of workflows on Cloud Computing infrastructures is an open challenge, that is receiving increasing interest. The Event-Driven paradigms can be profitably applied to orchestrating Cloud Workflow applications. The Orchestration Engine embodies such principles in a Serverless context, being designed to strictly follow Event-Driven Architectures and Event-Driven Microservices patterns. In this framework, the automatic management of the Serverless invoicing is handled by reactively scheduling the assignment of the Workflow tasks to the Cloud resources and by autonomously optimising the characteristics of those resources at runtime.

Event-Driven Architectures (EDA) are a standard paradigm for tackling complex computing problems that can be formulated as a set of independent business tasks, whose results are usually made available into an event bus (e.g. Kafka), where interested subscribers can consume those outputs. An typical approach based on EDA assumes that each application component is responsible for solving a specific task, and the integrated and orchestrated combination of those task-solving components can achieve any arbitrarily complex solution of the complete problem. Cloud Workflow orchestrators can be seen as Event-Driven Engines, built around an event-bus (typically, a message queue) and a central entity that subscribes and reacts to allocated events by delegating subtasks to independent worker services.

Graph 1 — Threshold-Triggered Mode Switching



Graph 2 — Cost vs. Confidence Trade-off



4. System Model and Components

An event-driven serverless orchestration framework enables autonomous cloud workflow optimization at runtime. The proposed system model comprises an orchestration engine and a cloud workflow composed of scheduled jobs. The orchestration engine, the driving component, responds to five different events. It executes or schedules cloud jobs in response to workflow execution context changes, schedules cloud jobs for faster completion based on cloud resources freed up by successful job executions, and attempts to mitigate transient resource capacity shortages in the cloud during workflow execution. Four different scheduling/optimization algorithms, combined with real-time reactive scheduling, realize autonomous workflow optimization. Experiments conducted on a real testbed consisting of an OpenStack cloud setup with various application workflows demonstrate the advantages of the proposed architecture.



Cloud workflows composed of several sequential and parallel jobs can take a long time to complete. The actual completion time of workflows may differ significantly from the estimated completion time because cloud resources can often go idle due to transient capacity shortages. Recent research has shown that cloud workflows can be executed more quickly and efficiently by searching for execution opportunities for jobs that were not yet scheduled based on the changing execution contexts of other cloud jobs. An event-driven architecture that employs serverless computing for implementation can make the process even easier and more efficient.

A. Orchestration Engine

The orchestration engine defined in this section supports the event-driven paradigms and models. This engine relies on both the built-in services of the serverless cloud and event-driven microservices at user-defined layers. Built-in services of the serverless cloud such as triggering events from user-defined thresholds on CPU utilization and running the monitoring and control function to switch on or switch off virtual machine instances are utilized through the scheduler logical layer. Whenever an event is triggered, the engine searches for candidate microservices based on the service registry to execute a user-defined task. If a nonbuilt-in user-defined task can execute in response to the triggered event, scheduled services for the event are executed without any waiting time; otherwise, the service is delayed until the service is ready. Events are searched from the event queue for execution, both periodically at user-defined intervals and in response to built-in events' triggers. The engine is commonly designed for all three levels, and the definition of services at each level solves the problem of overhead introduced with the execution of serverless functions.

The event-driven architecture closely resembles the publish-subscribe pattern. Services communicate through a message broker service. Services can subscribe to any events of relevance, acting both as publishers creating the events and as subscribers to the downstream services. Scaling and fault tolerance are entirely managed by the underlying service abstraction framework of the serverless cloud and by the publisher-subscriber pattern, thereby leveraging the benefits of the event-driven architecture. The service layer enables the automatic allocation of the required resources, and a user-defined threshold activates resource allocation for services at risk of degradation. The time required for the allocation of resources is usually longer than the time of service execution.

Hence, it is more suitable for services that are invoked less often and those that may take longer to complete.

5. Scheduling and Optimization Algorithms

The proposed event-driven serverless orchestration framework encompasses intrinsic scheduling and optimization algorithms that minimize execution overhead without user intervention. Utilized to ease the burden on the human actors in a cloud orchestration environment, scheduling algorithms typically use workload information to provide near-optimal solutions. Nevertheless, they can never reach a close-to-optimal solution without any workload information. The absence of workload and resource usage information undeniably increases the human effort for monitoring.

These drawbacks are addressed by implementing a real-time reactive scheduling algorithm that monitors the workload and resource usage in real-time within a certain threshold. Whenever this threshold condition is satisfied, it reacts and schedules the bits to the cloud to minimize the system execution overhead. Most of the existing smart workflow orchestration engines use complex and non-scheduled optimization algorithms, which introduces extra processing overhead. Therefore, a lightweight standalone optimization algorithm is designed. It adapts to rapid execution loading and provides periodic optimal configurations to achieve a fine balance between processing overhead and workflow execution time.

A. Real-Time Reactive Scheduling

Two classes of real-time workflows are modeled. In the first class, task arrival is triggered by external events, and values of the events are drawn from a known distribution. The objective is to schedule tasks with hard deadlines. These deadlines are quantised and defined by the existing of time slots in the calendar. A Task Pool Scheduling algorithm allocates resources to the incoming tasks at each event arrival. The optimal resource allocation strategy is studied using a queueing model. It requires the knowledge of the type of tasks and their arrival distribution. For each type of task, the distribution of resource intensive and time consuming tasks is used.



6. Conclusion

The event-driven serverless orchestration framework tackles multi-cloud workflows by intelligently coordinating cloud services and computing resources using event-driven paradigms. Existing approaches control workflows in a fixed manner without support for resource scheduling, adjustment, and optimization during task execution, which may lead to unexpected cost increases and service-level agreement violations for long-running workflows. Each proposal component builds on sound theoretical foundations and enables event-driven behavior. It is verified through simulation that by instinctively following the described theories, the event-driven serverless orchestration framework can automatically maintain QoS levels and optimize execution costs. The real-time reactive scheduling algorithm was also verified through cloudlet experimentation, with results matching theoretical expectations.

Cloud computing is emerging as the new computing paradigm of the new millennium. Cloud-based services allow users to create anything with complete creation freedom. Multicloud provides more options, giving users independence from a single service provider. Despite the advantages, the latency of calls to cloud services can complicate the construction of workflows especially for long-running tasks. Therefore, scheduling clouds using cloud services emerges as a new trend. Cloud service like AWS Lambda runs in response to events and manages computation resources. Event-driven programming is one of the leading programming paradigms today. Connecting it with cloud workflow execution thus provides a new approach.

REFERENCES

1. Secinaro, S., Calandra, D., Secinaro, A., Muthurangu, V., & Biancone, P. (2021). The role of artificial intelligence in healthcare: A structured literature review. *BMC Medical Informatics and Decision Making*, 21, Article 125.
2. Rajendran, S., Obeid, J. S., Binol, H., D'Agostino, R., Jr., Foley, K., Zhang, W., Austin, P., Brakefield, J., Gurcan, M. N., & Topaloglu, U. (2021). Cloud-based federated learning implementation across medical centers. *JCO Clinical Cancer Informatics*, 5, 1–11.
3. Granlund, T., Stirbu, V., & Mikkonen, T. (2021). Towards regulatory-compliant MLOps: Oravizio's journey from a machine learning experiment to a deployed certified medical product. *SN Computer Science*, 2, Article 342.
4. Gadi, A. L. (2019). Enhancing Vehicle Lifecycle Management through Integrated Data Platforms and IoT Connectivity. *International Journal of Engineering and Computer Science*, 8(12), 24973–24992. <https://doi.org/10.18535/ijecs.v8i12.4443>
5. Khan, N., Yaqoob, I., Hashem, I. A. T., Inayat, Z., Ali, W. K. M., Alam, M., Shiraz, M., & Gani, A. (2021). Big data: Survey, technologies, opportunities, and challenges. *The Scientific World Journal*, 2021, Article 712826.
6. Lu, Z., Qian, P., Bi, D., Ye, Z., He, X., Zhao, Y., Su, L., Li, S., & Zhu, Z. (2021). Application of AI and IoT in clinical medicine: Summary and challenges. *Current Medical Science*, 41(6), 1134–1150.
7. Abad, Z. S. H., Maslove, D. M., & Lee, J. (2021). Predicting discharge destination of critically ill patients using machine learning. *IEEE Journal of Biomedical and Health Informatics*, 25(3), 827–837.
8. Kodanda Rami Reddy Manukonda. (2021). Cybersecurity Threats Impacting Scheduling and Routing Software in Telecommunications. *Journal of Scientific and Engineering Research*, 8(12), 288–299. <https://doi.org/10.5281/zenodo.13254322>
9. Sun, Z., Yin, H., Chen, H., Chen, T., Cui, L., & Yang, F. (2021). Disease prediction via graph neural networks. *IEEE Journal of*



- Biomedical and Health Informatics, 25(3), 818–826.
10. Vučinić, M., & Luburić, R. (2022). Fintech, risk-based thinking and cyber risk. *Journal of Central Banking Theory and Practice*, 11(2), 27–53.
 11. Gill, S. S., Tuli, S., Xu, M., Singh, I., Singh, K. V., Lindsay, D., Tuli, S., Smirnova, D., Singh, M., Jain, U., & Buyya, R. (2022). Transformative effects of IoT, blockchain and AI on cloud computing: Vision, trends and open challenges. *Internet of Things*, 20, Article 100118.
 12. Mangalampalli, B. M. Intelligent Data Profiling for Healthcare Data Lakes Using AI-Enhanced Analytics.
 13. Motwani, A., Shukla, P. K., & Pawar, M. (2022). Ubiquitous and smart healthcare monitoring frameworks based on machine learning: A comprehensive review. *Artificial Intelligence in Medicine*, 134, Article 102431.
 14. Shi, S., Tse, R., Luo, W., et al. (2022). Machine learning-driven credit risk: A systemic review. *Neural Computing and Applications*, 34, 14327–14339.
 15. Zhang, A., Xing, L., Zou, J., & Wu, J. C. (2022). Shifting machine learning for healthcare from development to deployment and from models to data. *Nature Biomedical Engineering*, 6, 1330–1345.
 16. Dhiyanesh, B., Rameshkumar, M., Karthick, K., Radha, R., & others. (2023). Cloud computing and machine learning for analysis of health care data based on neuro fuzzy logistic regression. *Journal of Intelligent & Fuzzy Systems*, 44(6), 9955–9964.
 17. Aleksandrova, A., Ninova, V., & Zhelev, Z. (2023). A survey on AI implementation in finance, (cyber) insurance and financial controlling. *Risks*, 11(5), Article 91.
 18. Elragal, R., Elragal, A., & Habibipour, A. (2023). Healthcare analytics—A literature review and proposed research agenda. *Frontiers in Big Data*, 6, Article 1277976.
 19. Manukonda, K. R. R. (2020). Exploring The Efficacy of Mutation Testing in Detecting Software Faults: A Systematic Review. *European Journal of Advances in Engineering and Technology*, 7(9), 71–77.
 20. Muley, A., Muzumdar, P., Kurian, G., & Basyal, G. P. (2023). Risk of AI in healthcare: A comprehensive literature review and study framework. *arXiv preprint*.
 21. Wang, N., & Wang, K. (2022). Internet financial risk management in the context of big data and artificial intelligence. *Mathematical Problems in Engineering*, 2022.
 22. Xie, H., & Shi, Y. (2022). A big data technique for internet financial risk control. *Mobile Information Systems*, 2022.
 23. Wang, Y., & Wang, L. (2022). Internet financial data security and economic risk prevention for Android application privacy leakage detection. *Computational Intelligence and Neuroscience*, 2022.
 24. Mashetty, S. (2023). A Comparative Analysis of Patented Technologies Supporting Mortgage and Housing Finance. *Educational Administration: Theory and Practice*.
 25. Wu, Z., Zhou, Q., Wang, L., & Zhao, D. (2023). Research on internet financial risk control based on deep learning algorithm. *Soft Computing*.
 26. Yadav, S., Kaushik, A., & Sharma, S. (2022). Simplify the difficult: Artificial intelligence and cloud computing in healthcare. In



- EAI/Springer Innovations in Communication and Computing (pp. 101–124). Springer.
27. Rahimi, B., Safdari, R., & Mohammadi, A. (2022). Cloud healthcare services: A comprehensive and systematic literature review. *Transactions on Emerging Telecommunications Technologies*, 33(7), Article e4473.
 28. Kumari, J., Kumar, E., & Kumar, D. (2023). A structured analysis to study the role of machine learning and deep learning in the healthcare sector with big data analytics. *Archives of Computational Methods in Engineering*.
 29. Mangalampalli, B. M. Generative AI Applications In Healthcare Data Mart Design And Optimization.
 30. Large-scale data-driven financial risk management & analysis using machine learning strategies. (2023). *Measurement: Sensors*, 27, Article 100756.
 31. Kumar, S. S. (2023). AI-based data analytics for financial risk governance and integrity-assured cybersecurity in cloud-based healthcare. *International Journal of Humanities and Information Technology*, 5(4), 96–102.