International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Optimizing Serverless Processing with a Smart Hybrid Cloud Scheduler

Rajashekhar Reddy Kethireddy

Abstract: ***Serverless computing, driven by FunctionasaService FaaS, and edge computing are both expanding fields. While both offer significant benefits, they are often incompatible. This paper proposes a hybrid approach to combine the strengths of both paradigms using a heterogeneous edgecloud infrastructure based on KubeEdge. The proposed architecture enables serverless computing frameworks like OpenFaaS to operate across cloud and edge nodes, optimizing the scheduling of FaaS workloads to enhance performance and efficiency.***

Keywords: Hybrid Cloud, Orchestration, Serverless Processing, Edge Computing, FunctionasaService

1. Introduction

Cloud orchestration is the practice of integrating resources from both public and private clouds, including apps, APIs, and infrastructure, to build unified workflows. By utilizing cloud orchestration platforms, IT departments can coordinate the automation of cloud administration tasks across multiple teams and domains. By integrating automation into processes in multicloud and hybrid cloud environments, administrators can enhance the dependability and efficiency of cloud computing systems [1]. This makes it easier for administrators to manage these systems over time.

1.1 Why is cloud orchestration necessary

While microservices and containerization increase the flexibility of cloud computing, they also add complexity that can be challenging to manage. Organisations frequently find it difficult to coordinate their operations and business processes, decrease the number of provisioning errors, and restrict resource sprawl [2]. This is because hybrid clouds are composed of a large number of diverse components, which can be found anywhere from on - premise datacenters to nearby edge sites. Automated processes are now necessary for cloud environments to work properly. Without automation, it would be practically difficult to complete the numerous complicated activities that are involved in maintaining cloud systems. IT teams have the ability to use automation to carry out a wide range of cloud management duties, including the following:

- Maintaining and deploying servers.
- Load balancers, routers, and switches are all pieces of networking gear that need management.
- Locating the accessible storage area.
- Creating VMs, or virtual computers.
- Utilising application deployment.

The automation of these procedures can lessen the likelihood of errors caused by human intervention and free up resources, so providing businesses with additional time to develop and provide services that are beneficial to their customers. On the other hand, these benefits are lessened when automated processes are separated from one another [3].

1.2 Cloud orchestration unifies multiple automation implementations into cohesive workflows

1.2.1 Cloud automation vs. cloud orchestration

Many people use the terms cloud orchestration and cloud automation interchangeably, but they are, in fact, distinct concepts. The phrase "cloud automation" describes the steps taken to make cloud management operations run smoothly with little to no human intervention. In this way, processes become more scalable and repeatable respectively. An example of this would be automating server instances such that they terminate when a task is finished.

The process of coordinating several automated tasks into higher - order workflows is referred to as cloud orchestration. This allows individual tasks to collaborate with one another in order to fulfil a certain purpose or process. Coordination of new infrastructure deployments with corresponding changes to on - premises network routing and firewall rules is one such example. Updating the load balancer at the same time as the operating system is another good example.

While cloud automation describes the act of automating individual functions, cloud orchestration describes the linking of several automated tasks to improve IT efficiency [4]. The analogy of a musical orchestra and its conductor is sometimes used to try to illustrate this disparity. Cloud orchestration is like the conductor of a symphony, coordinating all the different instances of cloud automation.

It is the conductor's job to make sure that each musician plays their part at the right time and with enough force. This makes it possible for the whole orchestra to play in perfect harmony throughout. Automation, like a musician, can play its part well enough to complete certain tasks. Nevertheless, orchestration is crucial in cloud environments where automation serves several functions within an organisation. This ensures that different types of automated operations work together to complete larger processes. If a company automates cloud management tasks throughout their hybrid cloud environment, they will inevitably have to integrate these activities into more efficient automation workflows, which is a logical next step after cloud automation [5]. After automating tasks in the cloud, the next logical step is cloud orchestration.

1.3 Benefits of cloud orchestration

By employing cloud orchestration, it is possible to efficiently automate tasks such as workload distribution, resource allocation, and service delivery inside a cloud environment.

Regardless of where they are operating, workflows may be created with orchestration solutions that coordinate automation processes across infrastructure. This ensures that all actions linked to the workflow are completed in the correct order. Information technology teams may build, maintain, and manage cloud resources and the software components that make them up as one cohesive unit with the help of cloud orchestration tools. Afterwards, they can use a template to deploy these resources automatically and repeatedly. Consolidating disparate automation processes into streamlined end - to - end workflows can help organisations achieve several goals, such as reducing provisioning errors, improving communication between apps and infrastructure, and implementing governance standards throughout their hybrid cloud [6].

Nevertheless, unified automation platforms offer connection and administration features that standalone solutions lack. There are a variety of automation technologies available for use in cloud administration that can take care of common tasks. By providing a consolidated foundation, a unified automation platform allows an organization's teams to communicate more effectively, share automation assets and best practices, and maintain uniform processes.

1.4 How can Red Hat help

Organisations can streamline their cloud operations management, tracking, and optimisation with Red Hat® Ansible® Automation Platform. This platform eliminates the need to switch between several domain - specific tools. Tasks like application deployment, provisioning, and configuration management fall under this category.

With the help of the Ansible Automation Platform, businesses can standardise their operational framework across all cloud domains, processes, and roles and bring automation closer to their endpoints. The platform integrates pre - existing automation, configuration, and cloud tools and processes using a human - readable YAML language. Apart from that, it gives IT pros a plethora of configurable options for orchestrating automated activities, such integrating Ansible Playbooks or building workflows in automation controller platforms.

Both Red Hat and its partners have created more than 135 Certified Content Collections. With these bundles, you can get real compatibility and support for use cases across many vendors and clouds, including AWS, Google Cloud, Azure, and a whole lot more besides. Plus, these cloud platforms provide a trustworthy and expert - led approach to executing core operational tasks with Ansible certified content [7]. The following are just a few of the many critical areas of cloud management that can be enhanced with the orchestration capabilities provided by the Ansible Automation Platform:

- **Configuration.** Cloud infrastructure and configuration management with a focus on security is offered via the stable and maintained Ansible Automation Platform by Red Hat.
- **Security.** By utilising a carefully curated set of modules, roles, and playbooks, IT teams can orchestrate security systems to detect, investigate, and respond to threats automatically.
- **Application deployment.** Automating application deployment, creating reliable and repeatable installation and update procedures, and building repeatable processes will help with Day 2 operations.
- **Container deployment.** Automation and management of Kubernetes or Red Hat OpenShift® deployments, as well as scaling of containerised applications, are made possible with the help of Kubernetes operators and frameworks.

With a subscription to the Ansible Automation Platform, you can automate your business at a hybrid cloud scale with access to open - source innovation that is customised to your specific needs. You will also have access to all the tools, services, training, and support you need.

2. Literature Review

In line with the current trend towards microservices application development, new methods are appearing that take into account the merging of cloud and edge computing management frameworks, with the goal of providing unified administration of resources throughout the computing continuum [8]. When we talk about resources that could be located anywhere along the infrastructure's edge, cloud, or Internet of Things (IoT) segments, we're talking about the computing continuum. The goal is to provide solutions that can optimally handle the latencysensitive components of distributed applications, typically delivered close to the edge where they are consumed. When it comes to Quality of Service (QoS), these parts are very picky. A number of fields, such as augmented and virtual reality, emergency communications, disaster management, and remote healthcare, share the need to support distributed workflows powered by Machine Learning (ML) and to enable tight interaction between the IoT and edge computing nodes [9]. The development of mature serverless computing apps is a crucial step in deploying computing continuum systems [10]. Even if they might be linked to a bigger application or service chain, serverless computing allows for the independent launch and scaling of individual functions or microservices in response to user request flows.

Some of the functions may be computationally expensive and best provided by the cloud, while others may be delay - intolerant and best provided by the edge computing component of the infrastructure. There is an urgent need for efficient scalability solutions to facilitate the management of intermittent workloads [11]. Optimal orchestration of functions also requires solutions that allow for optimum use of resources without over - providing those resources. A lot of factors need to be thought about, such as energy usage and how to keep infrastructure providers' overall costs to a minimum.

Developing orchestration solutions that properly separate continuum resources from application developers is critical for optimally providing serverless applications over continuum resources.

This is necessary because, unlike in conventional container or VM - based deployments, resources are not pre - provisioned but rather made available in response to actual user demand. Distribution of resources over multiple clusters necessitates

management solutions that can handle multiple clouds, if not all of them. Application developers, application providers, and infrastructure providers can all work together more effectively when intent - driven techniques are used in conjunction with the appropriate abstractions to specify different requirements [13]. Developers can build apps with the ability to declare deployment preferences, requirements, and constraints in mind [14], application providers can describe SLOs in line with any Service Level Agreements (SLAs) they may have with clients, and infrastructure providers can make sure these SLOs are met and that the applications are provided to their fullest potential.

So, effective scheduling of resources is another major obstacle. Discovering the optimal mapping of application service functions onto scattered and dispersed computing, network, and edge resources is crucial for achieving application - and provider - related objectives [15]. For serverless computing systems in particular, improving the placement of functions is crucial to minimising the applications makespan, which is defined as the total time taken to process a sequence of functions (application/service chain) for its complete execution. When optimising the performance of the function chain and minimising the overall cost to infrastructure providers, it is vital to consider the cold - start time of the serverless functions [16].

In addition, one of the most important ways to reduce network latency is to schedule serverless functions close to IoT devices. However, while deciding where to put the function chains, it's crucial to think about how the underlying infrastructure uses its resources and how much energy it consumes [17]. In this paper, we offer an intent - driven orchestration technique for serverless computing applications executed throughout the computing continuum, taking into consideration these developments and obstacles.

A series of serverless functions represents the apps. To ensure the best possible delivery of serverless apps, an intent is stated. To save customers or application developers the trouble of explicitly defining the resources they need to reserve, we're thinking about using an intent - driven method to describe SLOs [18]. Application providers and developers can affect the execution of serverless functions and help achieve certain objectives by being aware of SLOs. For instance, various deployment strategies and runtime orchestration actions can be employed to attain goals such as achieving high performance for an application portion or reducing costs. It is also possible to establish a requirement to prevent cold starts and the resulting performance changes. The same holds true for geographical restrictions; for instance, data privacy concerns may necessitate the collocation of certain serverless operations. Deployment strategies and orchestration tasks that are relevant across the continuum can be derived from these SLOs by translating them into appropriate SLAs [20]. The second component has its own unique set of problems, such as the necessity for coordinated effort from various application providers and the need for centralised control of resources across many clusters. Improving the automation and distributed intelligence of the orchestration techniques in use must always be the primary goal.

3. Orchestrate Distributed Services with Serverless And Kubernetes

The proliferation of serverless architectures has resulted in a resurgence of interest in serverless workflows as well as an increase in their relevance. In the past, they were considered to be centralised and monolithic, but now they play an important part in the orchestration of cloud - based events and services. A vendor - neutral way of defining service orchestration did not exist until recently. This meant that vendors and their implementations were the ones developers had to rely on. After some consideration, we decided that a globally recognised standardised language was necessary for describing serverless operations. The current version of the Serverless Workflow standard is 0.5, and we're preparing this paper to introduce it. The goal of this project is to provide the means for everyone to build serverless workflow libraries, tools, and infrastructure that can model workflows on different cloud platforms.

a) About the Serverless Workflow specification

The Serverless Workflow specification lays out a common, declarative language for building workflows. It is possible for developers to use it to design stateless as well as stateful orchestrations respectively. The standard is a sandbox project that is being hosted by the CNCF Serverless Working Group and is being developed by the Cloud Native Computing Foundation (CNCF).

Hosting a workflow language that is not dependent on any particular vendor, is portable, and is pushed by the community is the primary objective of the Serverless Workflow project. Instead of proprietary specs, this language is based on standards. Our main emphasis has been on the coordination of distributed event - driven systems. When describing a workflow with the Serverless Workflow specification, you have the option of utilising either the YAML or JSON format formats.

b) Structure of the Serverless Workflow language

An illustration of the structure of the Serverless Workflow language may be found in Figure 1. This structure is composed of three primary components. Every component is built around previously established criteria.

Figure 1: Three main parts make up Serverless Workflow's language layout. .

In the next few sections, we'll talk about the three main parts that make up the structure of the Serverless Workflow language.

c) Part 1: Defining events

The CloudEvents specification is used by Serverless process because it lets you define events that can be sent or received while the process is running. The specification outlines a straightforward mapping that is one - to - one between the manner in which events are specified within the CloudEvents format and the manner in which you define them inside your workflow language definitions. In order to build one or more event - correlation rules, you can make advantage of the context characteristics provided by CloudEvents. Serverless Workflow allows you to reuse event definitions, allowing you to utilise them in several workflows simultaneously. You can see the difference between a CloudEvent definition and a Serverless Workflow specification of the same event in Figure 2.

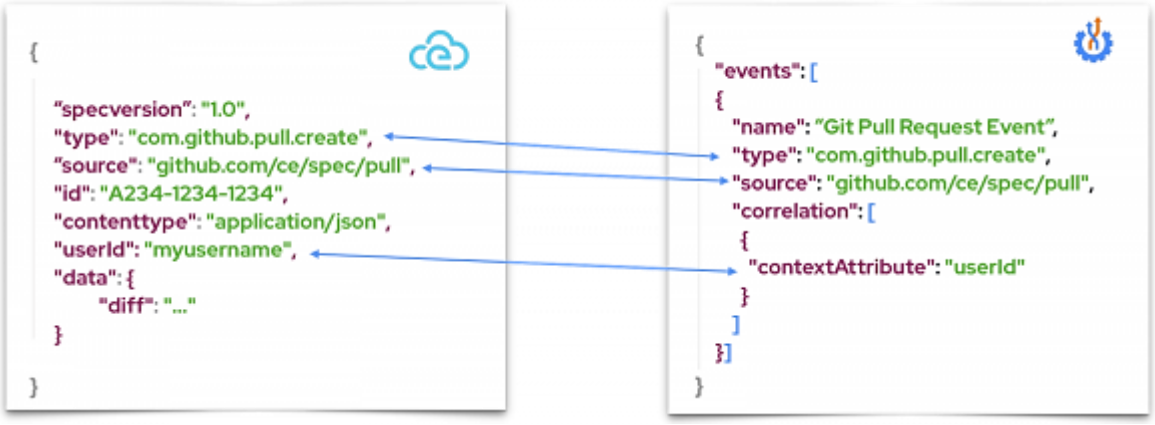

Figure 2: Compare the CloudEvents format with the Serverless Workflow format for defining events.

d) Part 2: Defining services or functions

During orchestration, Serverless Workflow uses the OpenAPI specification to define the services and operations that go along with them. Simply enter the path to the OpenAPI service definition (s) and identify the exact operation ID for the service operation you wish to invoke to define a service operation. The definitions of the Serverless Workflow service are displayed in Figure 3, along with associated OpenAPI definitions when applicable.

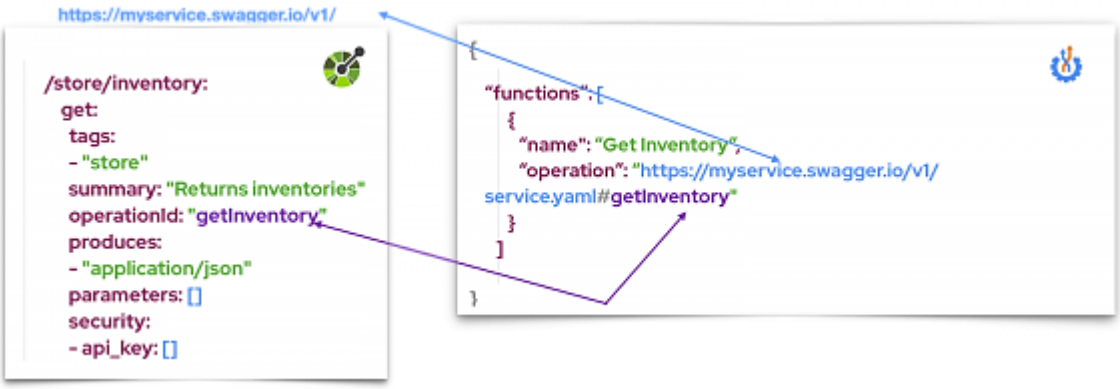

Figure 3: Definitions of Serverless Workflow services in connection to their OpenAPI descriptions.

Figure 4 shows that using Serverless Workflow, you can specify the invocation of RESTful and event - triggered services or functions.

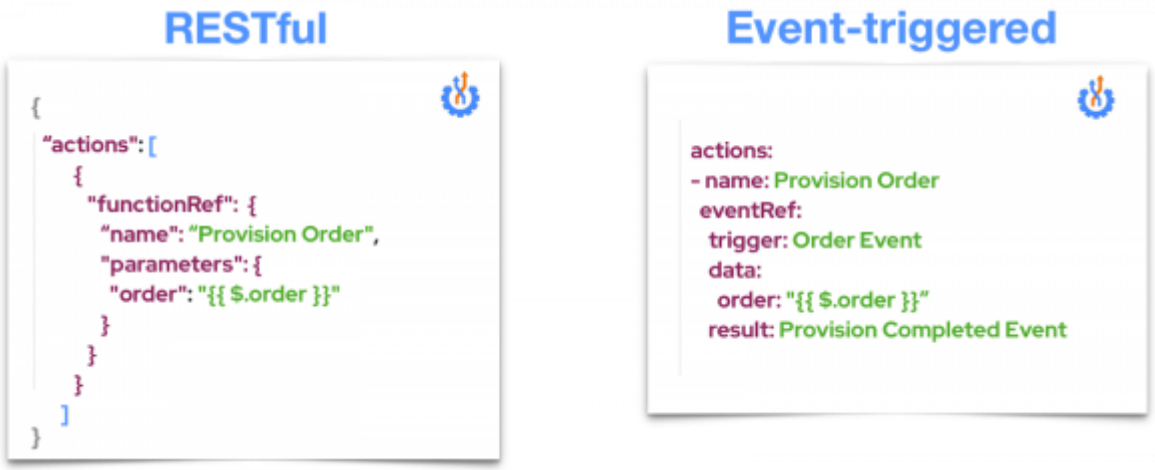

Figure 4: Depicting how RESTful and event - triggered services are invoked

e) Part 3: Defining the control - flow logic

The Serverless Workflow framework is responsible for developing pattern - based control - flow logic constructs that are used to define what will occur during the execution of workflows. Workflow states or phases, as well as the transitions associated with them, error handling, retries, data management, and other possibilities, can be defined by the user. As seen in Figure 5, Serverless Workflow's control - flow logic structures are illustrated. In order to guide the entire orchestration, you can utilise them to set up basic sequences or more complex structures like loops, retries, user interactions, or decision processes, or even parallel executions.

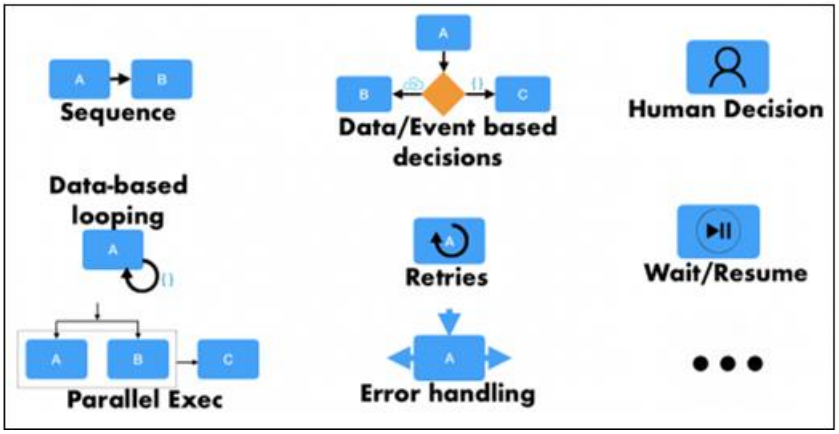

Figure 5: Serverless Workflow control - flow logic architectures

3.1 Additional components of Serverless Workflow

In addition to the workflow language definition provided in the specification for the JSON Schema definitions, Serverless Workflow also includes a variety of language extensions. The enhancements raise process parameters, which in turn improves orchestration's overall performance, cost, and efficacy. On top of that, Serverless Workflow provides software development kits (SDKs) for both Java and Go, with more SDKs in the works.

In addition to that, it comes with an extension for Visual Studio Code as well as an online editor, which offers helpful features like as code completions and diagram generation for workflow formats that are either JSON or YAML. The Serverless Workflow project is broken down into its component parts in Figure 6.

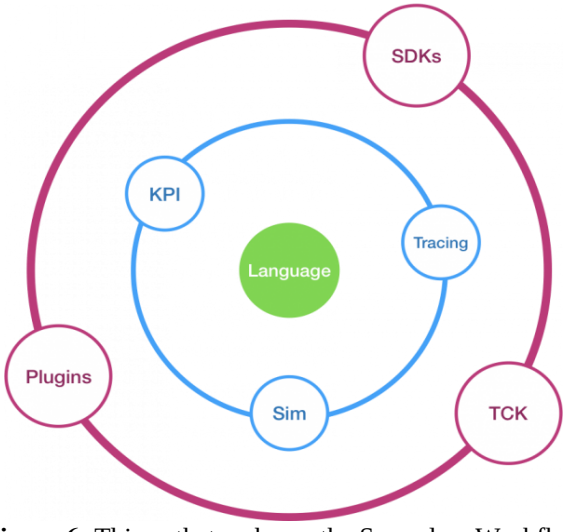

Figure 6: Things that make up the Serverless Workflow initiative

3.2 Use cases for Serverless Workflow

It is possible to utilise the Serverless Workflow specification language for a wide variety of use cases, such as the processing of payments, the analysis of data, the deployment of continuous integration, and many more potential applications. Several use cases are described in great length in the document that contains the specification's use cases. The flow diagram for an example of a use case is displayed in Figure 7.

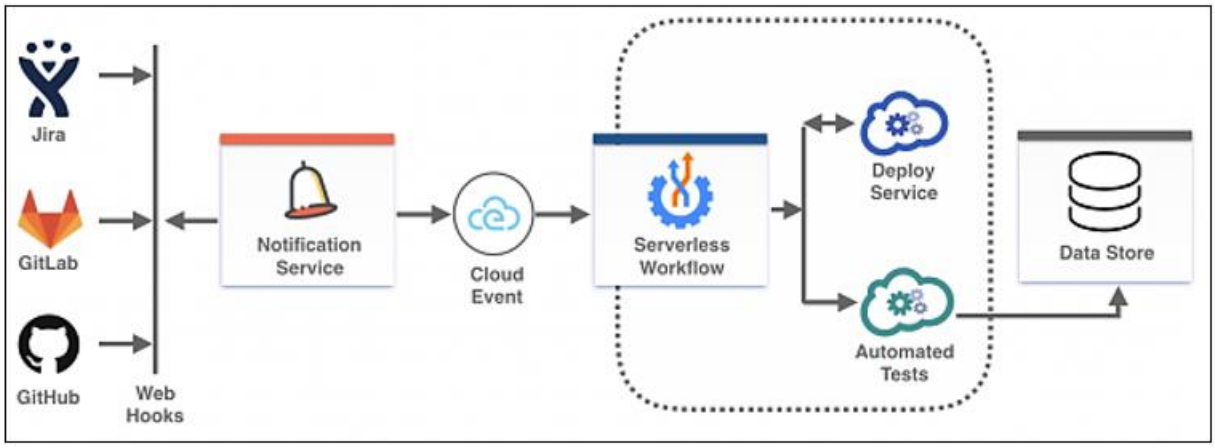

Figure 7: One possible application of Serverless Workflow.

Video demo: A Kubernetes use case
For container - based settings, Serverless Workflow is a top pick. Instantaneously, you can convert its service and event definitions into Kubernetes - specific structures like sinks and brokers. At KubeCon + CloudNativeCon North America 2020, we will be presenting a lecture that shows how to install Serverless Workflow for managing and orchestrating services running on a Kubernetes cluster.

3.3 Get involved with Serverless Workflow

Active community participation is essential to the success of the Serverless Workflow project. Everyone is welcome to attend and participate in our community gatherings that take place every week. The community gathering is a great venue for asking questions about the project. If you have any questions or concerns, you can also contact us through the Serverless Workflow community Slack channel or the GitHub source for the Serverless Workflow project.

Purpose:
The purpose of this research is to develop and evaluate a Smart Hybrid Cloud Scheduler HCS that optimizes the orchestration of serverless computing workloads across cloud and edge nodes.

Significance:
This research is significant as it addresses the challenge of integrating serverless computing with edge computing in a hybrid cloud environment, potentially leading to more efficient and scalable cloud services.

4. Conclusion

Serverless Workflow is a community driven, standards based, portable workflow language built on existing infrastructure. This research demonstrates the potential of a Smart Hybrid Cloud Scheduler HCS to enhance the orchestration of serverless workloads across cloud and edge environments. By integrating serverless and edge computing, the proposed solution can optimize resource allocation and improve the efficiency of cloud services. Further research and development could refine this approach, leading to broader adoption in various industries.

References

[1] Rosendo, D., et al. (2022). "Distributed intelligence on the edge - to - cloud continuum: A systematic literature review. " *Journal of Parallel and Distributed Computing*.
[2] Zafeiropoulos, A., et al. (2022). "Reinforcement learning - assisted autoscaling mechanisms for serverless computing platforms. " *Simulation Modelling Practice and Theory*.
[3] Cappanera, P., et al. (2019). "VNF placement for service chaining in a distributed cloud environment with multiple stakeholders. " *Computer Communications*.
[4] Dimolitsas, I., et al. (2023). "Time - efficient distributed virtual network embedding for round - trip delay minimization. " *Journal of Network and Computer Applications*.
[5] Son, J., et al. (2019). "Latency - aware virtualized network function provisioning for distributed edge clouds. " *Journal of Systems and Software*.
[6] Beloglazov, A., et al. "A taxonomy and survey of energy - efficient data centers and cloud computing systems. "
[7] Garí, Y., et al. (2021). "Reinforcement learning - based application Autoscaling in the Cloud: A survey. " *Engineering Applications of Artificial Intelligence*.
[8] Marotta, A., et al. (2017). "A fast robust optimization - based heuristic for the deployment of green virtual

network functions. " *Journal of Network and Computer Applications*.
[9] Dustdar, S., et al. (2023). "On distributed computing continuum systems. " *IEEE Transactions on Knowledge and Data Engineering*.
[10] Pujol, V. C., et al. (2023). "Edge intelligence—Research opportunities for distributed computing continuum systems. " *IEEE Internet Computing*.
[11] Russo, G. R., et al. "Serverless functions in the cloud - edge continuum: Challenges and opportunities. "
[12] Raith, P., et al. (2023). "Serverless edge computing—Where we are and what lies ahead. " *IEEE Internet Computing*.
[13] Mampage, A., et al. (2022). "A holistic view on resource management in serverless computing environments: Taxonomy and future directions. " *ACM Computing Surveys*.
[14] Metsch, T., et al. (2023). "Intent - driven orchestration: Enforcing service level objectives for cloud native deployments. " *SN Computer Science*.
[15] Zafeiropoulos, A., Fotopoulou, E., Vassilakis, C., Tzanettis, I., Lombardo, C., Carrega, A., Bruschi, R. "Intent - Driven. . . "
[16] Matrouk, K., et al. (2021). "Scheduling algorithms in fog computing: A survey. " *International Journal of Network and Distributed Computing*.
[17] Deng, S., et al. (2021). "Dependent function embedding for distributed serverless edge computing. " *IEEE Transactions on Parallel and Distributed Systems*.
[18] Li, Z., et al. "Help rather than recycle: Alleviating cold startup in serverless computing through inter - function container sharing. "
[19] Vahidinia, P., et al. (2023). "Mitigating cold start problem in serverless computing: A reinforcement learning approach. " *IEEE Internet of Things Journal*.
[20] Metsch, T., et al. (2023). "Intent - driven orchestration: Enforcing service level objectives for cloud native deployments. " *SN Computer Science*.