Review

Not peer-reviewed version

Serverless Architecture and Its Current State of the Art: A Systematic Literature Review

Ammad Ul Haq Farooqi , Omer Khalid , Muhammad Bilal *

Posted Date: 4 December 2025

doi: 10.20944/preprints202512.0219.v1

Keywords: serverless architecture; serverless computing; cloud computing; function-as-a-service; backend-as-a-service

 doi:10.20944/preprints202512.0219.v1

Review

Serverless Architecture and Its Current State of the Art: A Systematic Literature Review

Ammad Ul Haq Farooqi, Omer Khalid and Muhammad Bilal *

Department of Software Engineering, National University of Computer and Emerging Sciences, Islamabad 44000, Pakistan
* Correspondence: muhammad.bilal@isb.nu.edu.pk

Abstract

The serverless architecture has progressively become one of the most popular way to build and deploy applications now a days. This architecture allows the developers to focus on their code without worrying about managing the backend servers. By using abstraction principle, the serverless approach makes it very easier to achieve scalability, automatic resource management and cost efficiency via pay per use model. With increase in the usage of the serverless approach, this architecture has now expanded into domains such as the Internet of Things (IoT), high-performance computing, artificial intelligence and the large-scale cloud environments. With the expansion of the serverless computing comes the challenges as well which include performance challenges, reliability and maintenance challenges. This review examines fifty peer-reviewed studies to present a structured overview of the current state of serverless computing and it also organizes the existing work into a taxonomy that shows key developments across application domains, technical methods, data sources and limitations while also identifying open the research directions. We found out that there is a clear evolution from simple function orchestration towards more intelligent, workload-aware scheduling systems with improved cold start latency and hybrid deployments ability that span both cloud and edge infrastructures. However despite of these advances the recurring issues such as vendor lock in, limited debugging visibility, difficulties in managing state and unpredictable performance still pose a challenge to widespread adoption of the serverless approach. Finally we would also like to add that the review highlights several promising directions for future research as well which includes adaptive resource management, distributed serverless runtimes, AI-driven optimization and better support for heterogeneous hardware. All in all our work offers a consolidated understanding of the current progress and future potential of serverless computing approach.

Keywords: serverless architecture; serverless computing; cloud computing; function-as-a-service; backend-as-a-service

1. Introduction

In the last few years the serverless architecture or the serverless computing approach has emerged as one of the most efficient software architectures in the field of distributed computing. The reason behind this is the fact that the serverless approach allows the developers to focus more on application logic and code without worrying too much about the server management side of things [1]. The resulting code or the application that comes out of this approach is more modular, more event driven and you can say that its final shape is in the form of functions [2]. These functions are then deployed on the commercials platforms such as Amazon's AWS Lambda, Mircosoft's Azure Functions and Google's Cloud Functions [3,4]. These platforms then allocate computing resources under the FaaS (Function-as-a-Service) model or BaaS (Backend as a service) model to execute these functions and provide services back to the user [5–7]. This is how the serverless architecture or the serverless approach simplifies the development process but it also introduces complexities such as resource scheduling, cold starts and cost performance trade-offs which require some further attention with respect to research lens [8–10].

 doi:10.20944/preprints202512.0219.v1

As the time moves on the serverless computing has transformed the way the applications are built and deployed by transferring the responsibility of infrastructure management from developers to the cloud providers platforms [3,4]. Now instead of configuring the servers, managing the runtimes and handling the manual scaling, the developers can now focus more on the writing of the application logic and code while the cloud platform manages the execution and the scaling on their behalf [6,8]. This paradigm is now widely known as Function-as-a-Service (FaaS) model. This model brings several advantages such as automatic scaling, reduced operational effort and a pay-per-use cost model that improves efficiency [8,11]. However, as the serverless computing approach continues to expand into the diverse domains from IoT and; edge computing [12,13] to AI-driven analytics [14,15] and high-performance workflows [10,16]; its limitations have increasingly become more evident in the real-world scenarios [11,17,18]. Common challenges or limitations such as cold start latency [5,9,18,19], limited control over execution, debugging complexity [1,20], state management restrictions and unpredictable performance often complicate the system design and degrade the user experience. In addition to this many organizations struggle with vendor lock-in and the absence of standardized practices across platforms. These issues highlight that while the serverless computing approach simplifies the deployment while abstracting much of the infrastructure burden from the developers, it simultaneously also introduces a new layers of complexity at the architectural, operational and performance level that shows a growing need for a more indepth investigation and understanding [6,21,22].

While we were going through the finalized literature we found that there is a very large amount of research which explores many aspects of the serverless computing approach. Some of the literature takes a broad look at serverless computing to find out what role it plays in lowering the IT industry's carbon footprint [23] while some of it reveal that event-driven patterns, function chaining and API gateway integrations are central to achieving scalability. Additionally, the serverless APIs exhibit rapid elasticity, high fault tolerance and optimized resource utilization which conclude that the serverless architecture, when combined with robust design patterns, offers a good approach for scalable API deployment in dynamic digital environments [24]. Some of the literature takes a look at the evolution of fintech within the serverless computing paradigm, assessing scalability, performance and compliance, using case studies to examine its practical merits and ask where it falls short with regard to limitations [25] while other parts of it present a design of a cutting-edge, performance-focused, server-less computing platform that runs on Microsoft Azure, which was built in .NET, and used Windows container technology for function implementation [26]. There was also a study which had conducted the first detailed review of ten currently publicly available FaaS platforms which explored everything from their history, to their features and pricing to where they sit within the overall public FaaS landscape, before making a number of observations as to the state of the FaaS [27]. There was another study which had examined the technologies and characteristics that make serverless architectures possible and investigates the security and privacy issues that are unique to serverless computing using the real-world incidents to illustrate these concerns [28]. We had also found a study which discusses the importance of implementing robust security practices during the development and deployment stages of server less applications, including secure coding techniques, vulnerability scanning and runtime security controls [29] while there was also another study which explores the principles, benefits, challenges and practical applications of server-less computing in modern cloud architectures, highlighting its potential to drive efficiency and innovation [30].

In the literature we had also observed the current obstacles confronting serverless computing and they also explore potential avenues for future research to facilitate its deployment and utilization [31] also while examining the intersection of serverless computing and LLMs, particularly focusing on their application within the financial industry [32]. We had also observed a detailed analyses of the pros and cons of the serverless architectures bringing crucial insights into their ability to transform application scalability and cost savings when deployed in the cloud computing arena [33] and how the serverless computing has begun to transform the landscape of cloud computing by allowing developers to deploy applications without the burden of managing server infrastructure [34]. We had

 doi:10.20944/preprints202512.0219.v1

also observed a methodical review of related literature on the topic of serverless computing, to address the issue of the lack of compiling information on the state-of-the-art of the field of serverless computing in which we observed a comparison of the platforms and tools used in serverless computing and an extensive analysis of the differences[35]. We also went across a summary of findings and lessons which were learned from a series of research experiments that were conducted in the prior years in which we saw an argument that a careful attention must be placed on the promises associated with the serverless model which provide a reality check for common assumptions, and suggest ways to mitigate unwanted effects [36]. We had also come across a study in which we observed that the schedule of the individual invocations of functions, passed by a load balancer, was done in such as a way to minimize performance metrics related to response time [37]. So as the internet and the information communication technology has continued to evolve, the idea of computing has evolved with equal measure [38].

Some of the other aspects include the performance characteristics, the architectural models and applicability of the serverless approach across the different types of applications domains [11,21,39]. Many of these studies have also looked into the optimization strategies which help in reducing the cold start delays [5,9,18,19], while other studies emphasize on improving the resource utilization [40,41] and modeling of the system behavior under dynamic workloads [8,11,22]. Many researchers have also examined how the serverless platforms can be extended and integrated with the emerging domains such as edge computing [13,42,43], cloud-native data pipelines [3,4] and AI inference workflows [13–15] to enhance the scalability and responsiveness. Comparative analyses between the serverless approach, microservices approach and the traditional monolithic architecture have provided the valuable insights into the cost–performance trade-offs observed in real world deployments [3,4]. In parallel there are also some benchmarking tools and analytical models which have been developed to assess the performance consistency, to identify the bottlenecks and to support the configuration optimization [6,22,39,44] for the serverless approach. With these advances in the field of serverless computing there are a number of persistent challenges as well. These challenges include the debugging complexity [1,20], the performance variability across providers [21], the limited visibility into execution environments and the difficulties associated with stateful computation [11,20]. In the end, we can say that the current body of literature shows two facts. One fact is that the growing maturity of the serverless architecture acts as a practical computing paradigm and the other fact is the continued need to refine it in response to evolving computational demands.

Even though the serverless approach has made some good progress there are several gaps which remain which prevent us in fully understanding and optimizing the serverless systems [11,22]. The existing studies of taxonomy such as surveys, systematic reviews, and mapping studies etc., tend to focus on aspects such as cold start analysis [45], platform comparison [46], or performance and behavior [47], but they lack an integrated, multi-dimensional synthesis across different levels of perspective [48], [49]. Many of these secondary studies acknowledge limitations such as narrow dataset selection, inconsistent benchmarking, optimization strategies and an absence of cross-layer analysis connecting workloads, platforms [45,50,51]. They highlight the gaps in covering emerging domains like edge–cloud integration, AI-driven orchestration, and hybrid serverless deployments [48,52,53]. Unlike these studies, our review combines findings across fifty diverse papers published in the last five years to construct a unified taxonomy that links application domains, methods and techniques, data sources, limitations, and future work, offering a more thorough understanding of the state of the art. There are many studies which analyze the performance and some of them also propose isolated optimizations [9,18,19] as well but there are not many studies which offer a holistic or a overall view which connects the application domains, runtime behavior, performance trade-offs, system limitations and emerging solutions [6,22,44] and additionally we would also like to add that the current literature often evaluates the serverless systems under certain controlled conditions. Due to this we may not see it reflect the dynamic and diverse workloads observed in real-world deployments [21,54]. There are also limited discussions on how the serverless platforms should evolve to support more complex computational

 doi:10.20944/preprints202512.0219.v1

tasks, data-intensive pipelines and distributed environments spanning both the cloud and also the edge [13,42,43]. Our review study aims to address these gaps by extracting and combining findings from the fifty peer-reviewed studies to construct a unified taxonomy of the serverless architecture. By using the said taxonomy, this paper then organizes the analysis across the application use cases or the application domains, the methods and optimization techniques, the data sources, the performance and efficiency considerations and the recognized system limitations. By structuring the findings through the mentioned taxonomy structure, the review contributes a consolidating understanding of the present research direction and highlights the emerging opportunities for advancing the serverless computing.

This bar chart in Figure 1 summarizes the literature that has been published over the last 5 years from the year of 2020 to the year of 2025. We observed that the work reached it peak in the year of 2021.

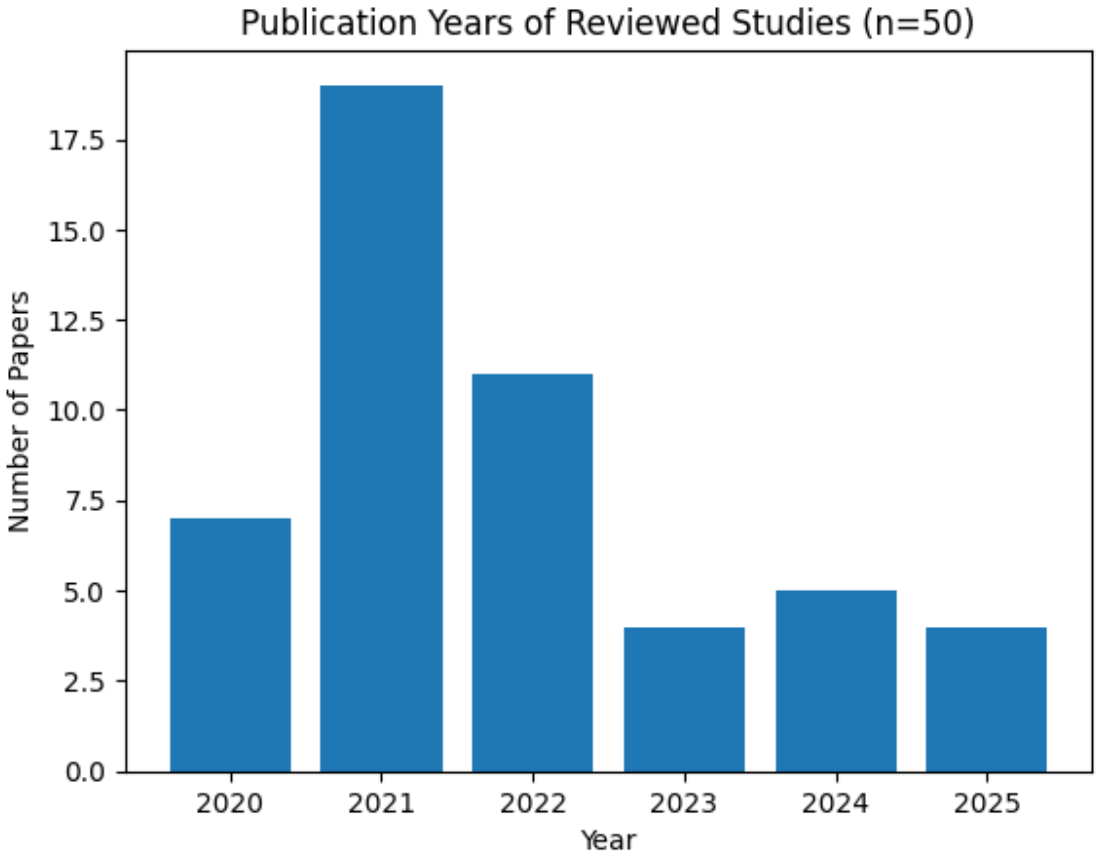

Figure 1. Number of papers distribution over the past years.

The rest of the paper is arranged in a way that reflects how the taxonomy developed during the review process. In the section 4 we have an outline that shows the materials and methods used to collect, evaluate, and synthesize the selected 50 studies. This includes the search strategies, inclusion criteria and the data extraction methods. In the section 5 we have the taxonomy diagram of serverless architecture itself and we also describes how the recent research themes align with the different branches of the taxonomy. In section 6 through 9 we analyze the reviewed works across four the key dimensions: application domains, methods and techniques, data sources used in evaluation and the system limitations reported in literature. In the section 10 we discuss the future research directions which were identified from the emerging trends and unresolved challenges. Finally, in the section 11 we provide the concluding remarks in which we summarize the key insights and the implications for ongoing research and the real-world adoption. By following this structured approach, the paper offers

 doi:10.20944/preprints202512.0219.v1

a clear and a comprehensive understanding of the serverless computing landscape and the direction of its future development.

2. Materials and Methods

While we were going through the literature we followed the PRISMA document which was released back in the year of 2020. The main purpose for using it was for guidance to ensure a transparent and a replicatable study selection [55][76]. The search keywords included: "serverless architecture", "FaaS", "function as a service", "serverless computing", "cold start", "autoscaling", "resource allocation", and "serverless deep learning". Searches were executed on IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, and ResearchGate between January 2020 and October 2025.

Each candidate paper was screened in three stages: title/abstract screening, full-text retrieval, and eligibility assessment. We removed duplicates and applied exclusion criteria (non-peer-reviewed sources, surveys when primary studies were required, regional low-quality outlets). From an initial set of 75 records, 25 were excluded as secondary or out-of-scope, leaving 50 primary studies for detailed extraction and analysis (see Figure 2).

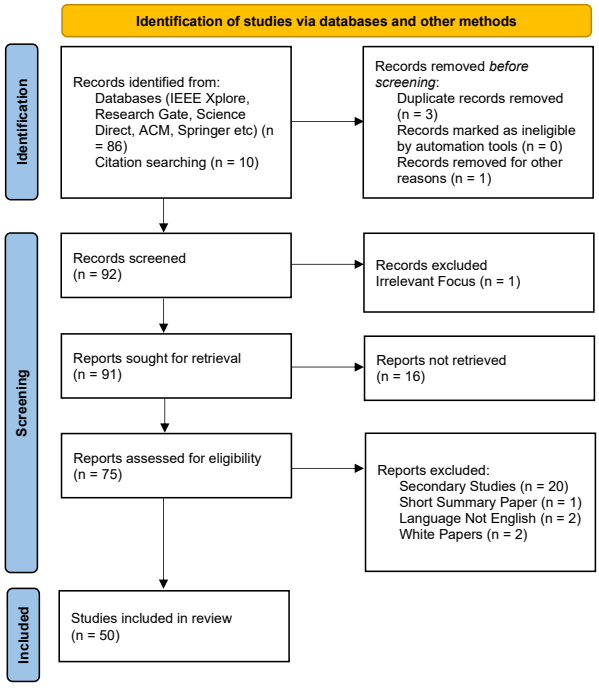

Figure 2. PRISMA Flow Diagram illustrating the study selection process.

2.0.1. Inclusion Criteria

Peer-reviewed journals or conference papers. Focus on serverless architecture, FaaS platforms, optimization techniques, performance evaluation, application case studies, or system limitations. Published between 2020 and 2025.

2.0.2. Exclusion Criteria

Non-peer-reviewed articles, blog posts, developer forums. Papers not providing technical depth or studies discussing only container-based or VM-based architectures without serverless relevance. Duplicate or incomplete publications.

The PRISMA diagram summarizes identification, screening, eligibility and inclusion steps and matches the counts reported in Figure 2.

 doi:10.20944/preprints202512.0219.v1

3. Taxonomy of Serverless Architecture and Its Current State of the Art

The taxonomy groups the literature into five main dimensions: (1) Application Domains, (2) Methods and Techniques, (3) Data Sources and Datasets, (4) Limitations, and (5) Future Works. This has been illustrated in the Figure 3.

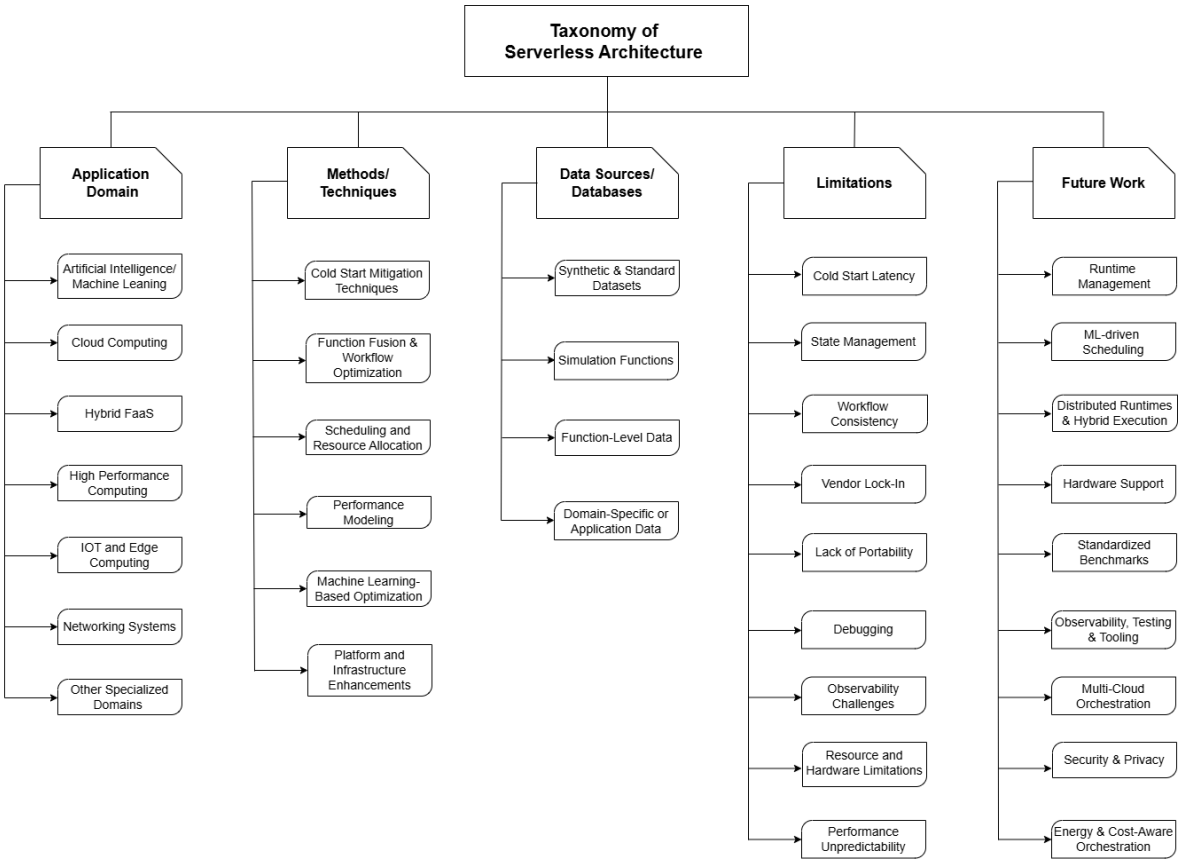

Figure 3. Proposed Taxonomy Diagram of Serverless Systems.

4. Application Domains

The serverless computing approach has demonstrated significant adaptability across multiple application domains by providing scalability and operational simplicity without the overhead of infrastructure management. Our study reflects its integration into AI/ML, (Internet of Things) IoT/Edge (Edge Computing), High-Performance Computing (HPC), Hybrid Cloud Systems and Networking environments. Each domain while it is benefiting from the flexibility that you get from the serverless approach, faces problems in shape of unique optimization and performance challenges that drive ongoing research forward.

4.1. Artificial Intelligence and Machine Learning

The serverless computing approach has gained a significant traction in the domains of AI and machine learning workloads. This is due to the elasticity and event-driven execution model present in the serverless computing approach. While going through the finalized literature we observed that many studies show that the serverless approach is very much suitable for the inference tasks where models can be triggered on demand, particularly in applications like document processing, image classification and real-time analytics [13–15]. For example we observed that several works show how the deep learning inference pipelines can be decomposed into functions which enables parallel execution while ensuring cost-efficiency [14,56]. However, the main challenges arise when dealing with large models or long-running training jobs, as the serverless platforms often impose execution time limits and the lack GPU acceleration [14,15]. To mitigate these constraints or issues the researchers have integrated the serverless architectures with external storage solutions like Amazon's AWS Elastic

 doi:10.20944/preprints202512.0219.v1

File System and hybrid provisioning models to maintain fast loading of model parameters [13,15]. ML-driven scheduling strategies, reinforced learning based cold start prediction and the workflow fusion approaches further illustrate how the AI methods are being used to optimize AI workloads on serverless platforms themselves. Overall, while the serverless computing shows clear strengths in inference scenarios and bursty AI workloads [14,15], continued innovation is needed to support full-scale training, hardware acceleration and distributed model handling [15].

4.2. Cloud-Native and Hybrid Serverless Systems

While going through the finalized literature we observed a portion of it also explores the application of the serverless architecture within broader cloud-native pipelines where the serverless functions act as workflow components which are then orchestrated alongside microservices, container platforms and traditional (Virtual Machines) or VMs [1,2]. These hybrid architectures allow developers to use serverless approach's scaling efficiency while maintaining the control over performance-critical operations [1]. The comparative evaluations across the monolithic, microservices and the serverless deployments show that the serverless often provides the cost and performance benefits under the variable workloads but it may become expensive under the sustained high-throughput scenarios [1,2]. Workflow orchestration tools like AWS Step Functions or Apache Airflow also influence cost-performance trade-offs, inspiring hybrid or adaptive orchestrator models [2]. As the organizations increasingly adopt the cloud-native practices, the serverless computing plays a crucial role in improving the modularity and deployment agility; though careful system design remains essential to balance cost, flexibility and reliability [1,2].

4.3. High Performance Computing (HPC)

The serverless computing approach is being explored in the high performance computing domain particularly for the parallel scientific workloads [10,16,56]. While going through the literature we found that those studies that implement the Monte Carlo simulations or data-intensive analytics show that thelarge-scale tasks can be decomposed into thousands of stateless functions which can then be executed concurrently [16,56]. This approach significantly reduces the computation time by exploiting massive horizontal scaling [10,56]. However the serverless platforms often lack guarantees for the low latency data movement or consistent compute performance; which are critical in HPC [10]. Cold starts, limited function runtimes and restricted memory can also limit compute heavy tasks. To overcome these limitations some studies adopt the container based extensions, specialized orchestrators or hybrid FaaS/CaaS deployments to handle longer running computations [10,56]. While the serverless computing approach may not replace the traditional super-computing systems it is emerging as a valuable complement especially for workloads that benefit from elastic burst scaling without dedicated cluster provisioning [10,16].

4.4. IoT and Edge Computing

The Internet of things (IoT) and edge computing environments benefit from serverless architecture due to their need for dynamic scaling and distributed event processing [7,12]. While going through the literature we found studies that highlight how the serverless platforms simplify the data handling in heterogeneous IoT devices by enabling the functions to run on-demand when a new sensor readings or events occur [14,43]. The serverless systems also reduce the operational overhead for the developers in building the remote or embedded systems since no consistent server management is required [43]. In addition to this, the serverless deployments at the edge platforms have shown to reduce latency compared to centralized cloud execution particularly for the applications which require real-time responsiveness such as smart city traffic systems, healthcare monitoring and robotics coordination [7,13]. However the bandwidth limitations, intermittent connectivity and the limited processing capabilities at the edge introduce new challenges [42,43]. To address these issues, hybrid architectures that combine Fog, Edge and Cloud layers have been proposed which ensure that the computation is dynamically placed based on energy cost, latency and workload type [7,12,13]. The literature

 doi:10.20944/preprints202512.0219.v1

reflects growing interest in these distributed serverless solutions though the standardized architectural patterns are still evolving.

4.5. Networking, SDN, and 5G

In the networking systems the serverless functions are used to dynamically manage the control plane logic, enabling the adaptive routing, efficient packet processing and energy-aware resource allocation [57]. The research works show that the integration of the serverless computing approach into the software defined networking or SDN environments demonstrates a noticeable improvement in scalability and energy efficiency. This is due to the fact that the network functions can be instantiated only when required [57]. Similarly the current research work also show that the serverless deployment at 5G edge nodes supports ultra low latency service delivery for mobile users [58]. However these architectures introduce coordination challenges between the network traffic controllers, orchestration systems and serverless runtimes [57,58]. Since the traffic prediction accuracy directly influences the cold start occurrences; the future systems are expected to rely more on intelligent (Machine Learning) ML-enhanced workload forecasting for real-time adaptation [57].

4.6. Other Specialized Domains

Beyond the mainstream application areas the serverless computing approach is also adopted in the fields/domains of robotics, geospatial data analysis, digital health data, financial transaction processing and distributed logging systems [21,59–61]. These domains value the serverless architecture for its modularity, simplified deployment and elasticity [59,60]. However the domain specific limitations often force the need for specialized optimization strategies such as improved data streaming reliability for robots or enhanced encryption protocols for healthcare data [21,61]. These studies demonstrate the versatility of the serverless computing while also reinforcing the need for customization and integration with complementary computing models [59,60].

5. Methods and Techniques

The research work on the serverless architecture spans across a wide range of methodological approaches that are aimed at enhancing performance, scalability and reliability. The studies reviewed can be grouped into six major categories: cold-start optimization, function fusion and workflow orchestration, resource scheduling and load balancing, performance modeling and benchmarking, machine learning driven optimization and platform extension/enhancement. Each sub area contributes towards reducing the latency, improving the resource efficiency and helps in achieving a predictable performance in Function-as-a-Service (FaaS) environments.

5.1. Cold Start Mitigation Techniques

One of the most frequently discussed problem in the domain of serverless architectures is the problem of the cold start latency. This problem occurs or happens when a function is invoked after a period of inactivity and the platform such as google, amazon or microsoft etc must initialize a new execution environment for it so that it may get executed [9,18]. While going through the finalized literature we found out that there are various mitigation strategies for it. These strategies include container prewarming, lightweight sandboxing, runtime caching and predictive invocation models [17,49] & [19]. In addition to this we would like to add that some of the research works use the machine learning process to learn and anticipate usage patterns for functions and then with that information those functions which were frequently accessed were proactively kept in a “warm” state [62]. While some works tried to overhaul the platform components to reduce the initialization overheads or they introduced the “adaptive cold start policies” based on the workload behavior [18]. All in all we would like to say that these approaches or techniques do help in reducing the variation in the response time of a function but they often increase the platform resource usage and this increases the cost and deceases the cost efficiency of the serverless platforms. This highlights the trade-off between performance consistency and cost efficiency of the serverless platforms [9,18].

 doi:10.20944/preprints202512.0219.v1

5.2. Function Fusion and Workflow Optimization

Here the function fusion refers to combining multiple small serverless functions into fewer but larger execution units to reduce latency which is caused by inter function communication and state transfer [9]. This strategy can help in improving the performance for workflows that have tightly coupled tasks or sequential dependencies [14]. However the fusion reduces modularity and this can reintroduce monolithic behavior if applied excessively [9]. Now in order to use function fusion in such a way that it does not brings excessive monolithic behaviour into the system we observed that in recent works there is proposal for selective or dynamic fusion techniques that analyze the call graphs, execution times and data dependencies before determining which functions should be merged [14]. All while for the work flow optimization we observed that some research works propose the use of workflow optimization methods which involves the caching of the intermediate results, streamlining of the data flow across functions and using event orchestration frameworks to coordinate complex pipelines efficiently [1,2].

5.3. Scheduling and Resource Allocation

As the serverless systems are highly dynamic we observed that effective scheduling is crucial for the managing of concurrency and resource utilization [40,63]. Some studies explore the scheduling mechanisms that account for workload intensity, network latency, memory demands and cost constraints [64,65]. While some approaches rely on heuristics based schedulers and some other approaches use the reinforcement learning and predictive models to assign workloads adaptively [62,63]. While going through the literature we observed the importance of edge-aware scheduling where the tasks are placed closer to data sources to reduce latency in (Internet of Things) IoT or real-time applications [13,65]. We also observed that the scheduling remains a complex challenge due to unpredictable workload fluctuations and platform transparency limitations [40,63].

Table 1. Categories of Data Sources Used in Serverless Computing Research.

Category	Examples	Purpose / Usage	Advantages	Limitations
Synthetic, Simulated Workloads And Benchmarks	COCOA [5], SCOPE [44], FaaSDOM [6], JMeter [4,13,62], custom load generators [14,22]	Stress testing scaling, concurrency, cold starts	Highly configurable	May not reflect realistic user behavior
Function-Level Execution Traces	AWS Lambda logs [3,4,11,56], Azure invocation traces [3,6,66]	Modeling runtime behavior, workload prediction, scheduling	Realistic and platform-specific	Limited access due to privacy/availability constraints
Domain-Specific Data	IoT sensor streams [7,12,67], network traffic logs [58,60], medical imaging data [7], video frames [43,59], NLP corpora [44,68], MNIST [13,15,56,63], EMNIST, LIBSVM, EPSILON [56]	Evaluating serverless applicability in real applications	Practical value and deployment feasibility	Harder to reproduce and standardize for comparison

5.4. Performance Modeling and Analytical Frameworks

We observed that there are several studies which developed the analytical models to evaluate the serverless performance under the varying conditions [8]. Techniques such as Layered Queueing Networks (LQN) and approximate concrete execution are used to simulate the request handling, function scaling and waiting times [8,69]. Benchmarking frameworks like FaaSDOM, SCOPE and COCOA provide structured ways to measure cold start latency, concurrency behavior and throughput across different cloud providers [6,44] and [5]. These models and tools help the researchers and the practitioners in making informed decisions about architecture design, cost estimation and performance

 doi:10.20944/preprints202512.0219.v1

tuning [6,8]. However the benchmarking process itself still varies widely across studies and this makes the cross-comparison challenging thing to do [5,44].

5.5. Machine Learning-Based Optimization

Across the literature we observed that there is growing area of research that integrates the machine learning into serverless orchestration and optimization [62,70]. We observed that some studies apply the machine learning techniques such as SVMs, KNN, Logistic Regression, LightGBM and Deep Reinforcement Learning to perform tasks such as workload prediction, autoscaling decisions, bottleneck detection and cold start anticipation [62,63]. While in other studies we saw that there are ML-driven control loops that enable adaptive decision making that improves the performance without the manual configuration [63,70]. However we also observed that incorporating the ML models introduces a new overhead and this requires reliable telemetry and monitoring which themselves are non-trivial challenges in serverless environments [44,69].

5.6. Platform and Infrastructure Enhancements

In the current literature we also saw that there are some research works that focus on modifying or extending the underlying serverless platforms to introduce features like GPU support, accelerated networking, enhanced state management and multi-runtime orchestration [10,15]. We also observed that systems such as Hybrid FaaS systems and container-augmented architectures provide more flexibility for long-running or resource-intensive workloads [10,56]. To conclude this section we will add that emerging work also explores integrating the serverless computing with blockchain, edge platforms and high performance distributed fabrics [7,13,57]. This suggests that the serverless systems may increasingly function as core components in broader distributed ecosystems [13,15].

6. Data Sources/Datasets/Databases

Evaluating the serverless systems require diverse datasets and workload sources to test the performance, latency behavior, scalability and application suitability. Across the reviewed studies we have identified three major categories of data sources: synthetic and workload-simulation dataset and benchmark., function-level execution traces dataset and domain-specific dataset. Each category depending on the application's domain and focus of the study plays a distinct role in testing and characterization of the serverless architectures.

6.1. Synthetic, Simulated Workloads and Benchmarks

Synthetic workloads remain the dominant method for stress-testing serverless behavior under configurable concurrency and request patterns. Benchmarking frameworks including COCOA [5], SCOPE [44], FaaSdom [6] and ServerlessBench [21,39] emulate invocation bursts, latency spikes, and scaling saturation to observe platform elasticity. Such workloads allow precise control of request rates, enabling measurement of cold-start frequency and auto-scaling response times [5,6,21,44]. However, purely synthetic testing may overlook cross-layer bottlenecks such as external I/O latency or multi-tenant interference [21,39].

6.2. Function-Level Execution Traces

We observed that the execution traces collected from AWS Lambda, Azure Functions or open research datasets capture the real invocation behaviors which offers insight into function duration, memory allocation and idle-time distribution [6,44]. We saw that empirical trace driven analyses by Mahmoudi and Khazaei [21,22,39] modeled runtime variability to develop predictive autoscalers, while Nguyen et al. [19] combined real-trace feedback with model-predictive control for proactive scheduling. Such datasets are very important for accurately modeling how the systems perform in the real world but the researchers often can't get them because of privacy rules and limited access to system monitoring data [21,39].

 doi:10.20944/preprints202512.0219.v1

6.3. Domain-Specific Datasets

Several studies use application-specific data sources to evaluate serverless performance in practical environments. For example, Pakdil and Çelik [59] utilized geospatial datasets for map-processing workflows. Ouyang et al. [12] applied IoT sensor streams for secure edge analytics and Nishimiya and Imai [71] used robotics telemetry to assess control-loop latency. Healthcare and financial systems have also employed domain datasets for privacy-aware orchestration [42,63]. These specialized datasets demonstrate the flexibility of serverless computing but are harder to reproduce or standardize for comparison.

This pie chart in Figure 4 highlights the nature of the datasets that is used in the finalized literature that has been published over the last 5 years from the year of 2020 to the year of 2025. We observed a distributed usage with respect to nature of dataset spanning across from domain specific to function level traces and synthetic type of datasets.

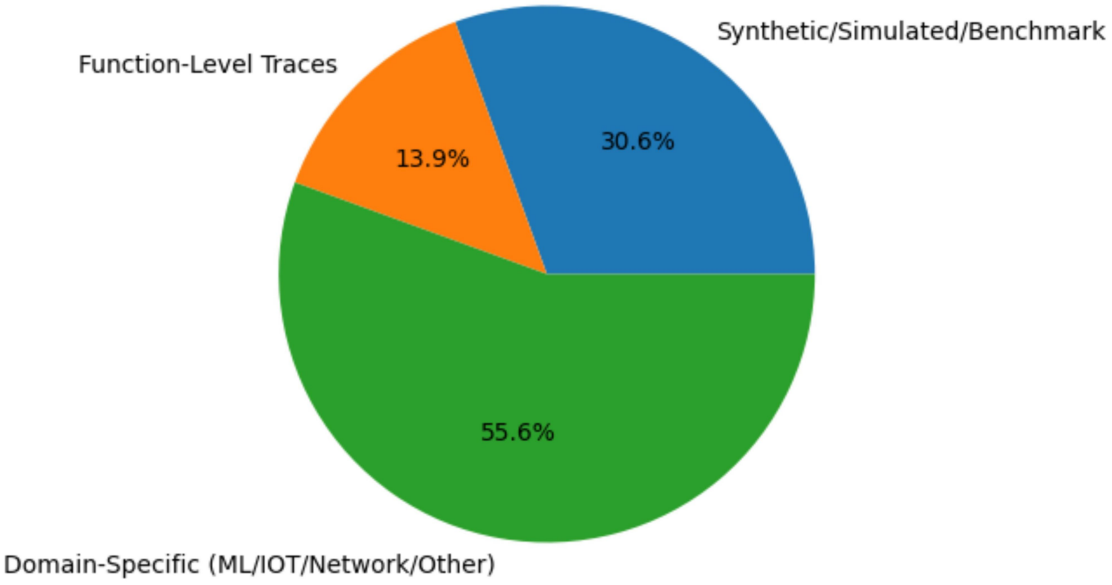

Figure 4. Distribution of datasets.

7. Limitations

While the serverless computing has advantages, it also has a number of disadvantages or limitations as well. These appeared across the recent studies as we were going through the literature. We observed that one of the main limitations of the serverless systems is the cold start latency in which the function initialization delays can slow down performance of the entire system.

We saw that there are issues such as higher latency and extra coordination overhead due to the reliance on the external services. We also saw that there is a vendor lock in challenge in which moving an application from one provider to another can be costly and time consuming. We also observed that the debugging and monitoring in serverless environemnt has its own challenges.

We would also like to add that the resource and execution constraints such as memory limits, maximum execution duration and lack of GPU support restrict the types of workloads that serverless platforms can efficiently handle. These limitations highlight that while the serverless computing simplifies the deployment and scaling side of things but it still requires a careful design in terms of trade offs and additional tooling to support complex and performance critical applications.

 doi:10.20944/preprints202512.0219.v1

7.1. Cold Start Latency

The cold start problem remains the single most cited bottleneck in serverless systems [19,62,72,73]. We observed in the literature that when the function containers sit idle for too long. Then they are need to be reactivated before handling the new requests. This often causes the unpredictable delays which produce latency spikes. The said latency spikes can seriously affect performance sensitive workloads such as the real time inference or the IoT event processing [17,18]. Now to overcome these issues we saw in the literature that there are techniques such as pre-warming and model predictive scheduling. These techniques can reduce the problem of cold start [62,72,73] to some extent but their main down side is that they also add platform overhead. This shows an current trade off between the responsiveness and the cost [9,17,60]. We also observed that the frameworks such as COCOA [5] & SCOPE [44] have started to measure and model this balance of responsiveness and cost. Along with this we had also observed that the main challenge of maintaining the cold start times under 100 ms mark is still unsolved [14,19,41].

7.2. State Management and Workflow Consistency

From the literature we observed that the serverless systems are essentially stateless due to which developers must rely on external storage systems such as S3, DynamoDB or (Elastic File System) EFS to manage persistence [4,22]. This dependency on external services often leads to a higher data access latency and added coordination overhead. This is especially valid in the case of workflow orchestration and AI/ML pipelines [11,14,21]. The research projects like Lambdata [74] and function fusion models [9] have made the progress towards easing this problem. These projects propose improvements in the transfer of intermediate state but achieving the full transactional consistency across the distributed functions still remains an open challenge [8,14,21].

7.3. Vendor Lock-In and Lack of Portability

We observed in the literature that the differences in runtime environments, API designs and configuration formats across the major cloud providers often leads to an issue know as a vendor lock-in. We observed that this issue continues to be a major obstacle for multi cloud adoption [4,10,22]. In practice moving the applications between platforms such as AWS Lambda, Azure Functions and Google Cloud Functions usually requires a good deal of refactoring and manual adjustment [22]. We also saw that there are open source projects such as OpenFaaS and Knative which have made progress toward standardizing deployment workflows [6,54] but true interoperability is still limited by the provider's proprietary runtime behavior and unique billing model [22].

7.4. Debugging and Observability Challenges

In the debugging side of things we saw that in the serverless systems this process continues to be a major challenge. This is mainly because of the fact that the functions run in short lived, opaque containers which have a very limited runtime visibility [1,4,20]. The logs and the traces that are available at time don't provide enough details and this makes it hard to pinpoint the root cause of issues or to fine tune performance [6,20,39]. Through De Silva and Hewawasam [1] we found that even the automated testing frameworks run into such difficulties when they are trying to inspect failures at the invocation level. This is particularly the case when the system scales up in parallel with other modules. So all in all these limitations point to the need for the better observability tools and more fine grained telemetry in future FaaS platforms [1,4,20].

The severity graph in Figure 5 provides a clear visualization of the dominant challenges found within the serverless computing domain. As illustrated, the majority of these issues are technical in nature, with cold-start latency emerging as the most significant and recurring concern.

 doi:10.20944/preprints202512.0219.v1

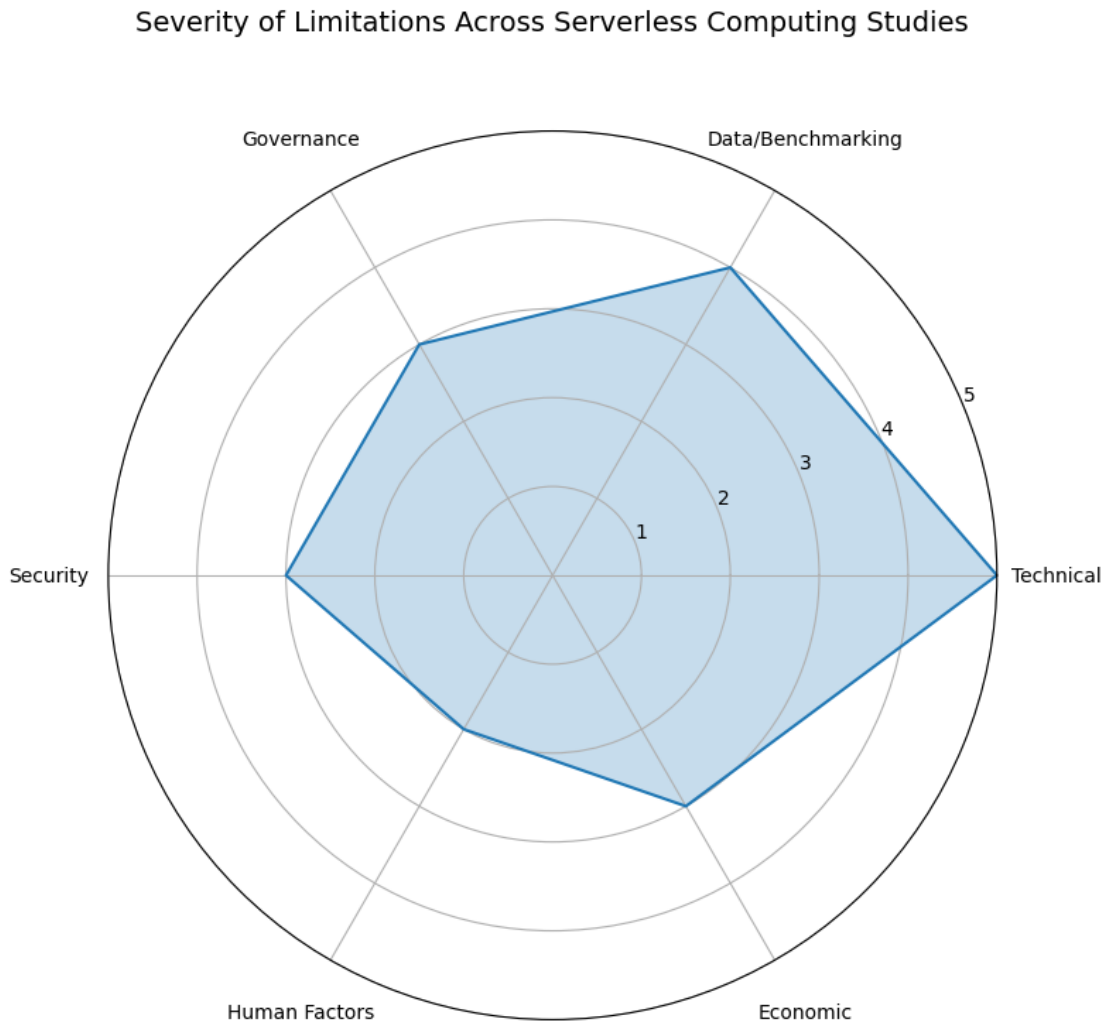

Figure 5. Core problems distribution in the Serverless Domain.

7.5. Resource Constraints and Hardware Limitations

In the literature we observed that many of the serverless environments still face strict limits on critical resources such as the memory, CPU and execution time and in most cases these environment do not provide access to GPU acceleration [15,22,43]. Because of these restrictions we saw that running and executing compute intensive workloads such as deep learning training and large scale scientific simulations still remains challenging [10,15,43]. Some of the researchers have started to use hybrid designs that rely on the container extensions or they offload parts of the work to edge devices [10,15,43]. While these approaches can ease the restrictions to some extent but they often add extra layers of orchestration that push overall costs higher [22].

7.6. Performance Unpredictability

We observed that the performance of the severless systems can vary significantly between the invocations. This is because of the multi tenancy and opaque scheduling that is observed in the platforms which providethe serverless services. Benchmarking studies such as COCOA [5], SCOPE [44] and FaaSdom [6] have consistently shown performance variance of over 25–40% even for identical workloads. This level of fluctuation makes it difficult to estimate costs accurately, enforce (Service Level Agreements) SLAs or maintain consistent (Quality of Service) QoS guarantees [22,39]. Some machine learning based predictive schedulers [62] have helped to reduce some of this instability but fully deterministic performance remains out of reach.

 doi:10.20944/preprints202512.0219.v1

8. Future Work and Research Directions

After going through the literature we believe that there is still a lot of potential for serverless computing to grow, particularly in the performance, flexibility and practical use side of things. One ongoing challenge in the serverless computing is dealing with cold starts and improving how the resources are used. Some recent methods such as adaptive pre warming, lightweight isolation and faster runtime environments have shown encouraging progress in reducing the startup delays. In parallel there is also an increase in interest of adding better support for GPUs (Graphical Processing Units) and TPUs (Tensor Processing Units) since these accelerators are becoming essential for running modern, compute intensive machine learning and data processing tasks. We observed that there is a worthwhile direction of hybrid orchestration models that mixes serverless, containers and edge resources making it easier to run applications across distributed systems.

As we move towards the end from the review we believe it could be understood that better benchmarking standards are needed so that different platforms can be compared fairly. Having richer datasets and access to shared execution traces would make it easier for us and our fellow researchers to study the workload patterns and autoscaling in more detail manner. At the same time we we think that integrating AI driven decision making into the workload prediction, scheduling and automated performance tuning could make these systems far more efficient. There is also a real potential in connecting serverless platforms with newer technologies like blockchain, quantum runtimes and distributed inference engines. Last but not the least dealing with vendor lockin through open deployment tools and cross platform orchestration frameworks will be keys to making serverless a truly portable and sustainable platform in the long run.

8.1. Adaptive and Predictive Runtime Management

Another observation that we made was that developing an adaptive pre warming and predictive warm pool management that can balance responsiveness and costs remains critical. So prior work using the capacity planning, model predictive control and reinforcement learning shows a strong potential for controllers that proactively keep the right functions warm based on learned invocation patterns and cost constraints [5,19,73]. Based on the current state of literature we think that the future work in serverless domain should focus on the lightweight models that operate with sparse telemetry (limited monitoring data such as system performance metrics, logs, traces etc) and on hybrid policies that trade a small, provable resource overhead for much lower cold start latency.

8.2. ML-driven Scheduling and Autoscaling at Scale

Machine-learning schedulers that generalize across workloads and providers are an active future work direction. Efforts such as FaaSRank and RL-based orchestration show promise, future research should evaluate transferable ML policies (trained on diverse trace data) and robust online adaptation to mitigate concept drift in invocation patterns [14,15,62]. Benchmarks of ML controllers under realistic noise and multi-tenant interference are required before production adoption [75].

8.3. Distributed Serverless Runtimes and Hybrid Execution

To support data-intensive and long-running workloads, research should further the design of distributed FaaS runtimes and hybrid FaaS/CaaS orchestration layers that combine fast elastic functions and long-lived containers seamlessly [10,14,40]. This includes consistent state-handoff mechanisms, placement algorithms aware of data locality, and lightweight RPC state channels to reduce function-to-function data transfer costs.

8.4. Heterogeneous Hardware Support (GPU/TPU/Edge Accelerators)

Expanding serverless to support hardware accelerators is essential for large ML and HPC workloads. Studies demonstrating GPU-assisted and edge-accelerated pipelines suggest work on transparent accelerator scheduling, GPU pooling strategies, and cost-aware accelerator tenancy models is

 doi:10.20944/preprints202512.0219.v1

needed to make FaaS viable for ML inference and some classes of training tasks [14,15,43]. Research should address isolation, cold-starts for accelerator contexts, and pricing models.

8.5. Standardized Benchmarks, Shared Traces, and Reproducibility

The community should converge on standard benchmark suites, shared invocation-trace repositories, and workload taxonomies to make results comparable across studies and providers [6,39,44]. Future work should produce canonical trace releases (anonymized) and standardized workload mixes that include bursty, diurnal, and adversarial patterns to stress test autoscalers and schedulers.

8.6. Observability, Testing, and Development Tooling

Improved observability primitives, deterministic local testing, and fault-injection frameworks are required so developers and researchers can reproduce, trace, and debug distributed functions at scale [6,20,39]. Research should explore lightweight distributed tracing standards for ephemeral functions and debuggers that can reconstruct causality across fused or dynamically re-partitioned workflows.

8.7. Portability, Multi-Cloud Orchestration and Interoperability

Mitigating vendor lock-in requires portable deployment artifacts, multi-cloud orchestration layers, and standardized runtime semantics that hide provider-specific behaviors [6,21,54]. Research can target provable migration tooling, semantic compatibility layers, and hybrid scheduler APIs that allow transparent failover and cost-aware cross-provider placement.

8.8. Security, Privacy and Federated Serverless Patterns

We have observed in the literature that the security and privacy continue to pose significant challenges for event-driven and serverless architectures. Based on our observation in the current literature we think that promising areas of exploration include privacy preserving serverless federated learning, attested execution for FaaS and lightweight capability models that manage how the functions are invoked and how data is accessed [7,63]. To make some real progress these solutions need to be studied and tested under realistic threat conditions and evaluated with practical performance goals in mind.

8.9. Energy-Aware and Green Serverless Scheduling

We observed that the energy aware scheduling was another area highlighted in the future research section of the literature. This concept included ideas such as energy fungibility, along with carbon aware placement strategies and it holds great promise particularly for edge cloud systems where energy costs and battery limitations are important factors [57,64]. We think that the future work should focus on quantifying the trade offs among energy consumption, latency and cost and on developing schedulers that are optimized for the sustainability metrics.

8.10. Economics and Cost-Aware Orchestration

Based on the observations made from the current literature we think that better cost models and cost aware orchestration methods that consider function billing, data egress and accelerator pricing could help operators and users make smarter deployment choices within realistic budgets [4,8,41]. Similarly we think that exploring ways to integrate pricing models into autoscaling policies especially those using spot or discounted resources could make these approaches more practical in real-world use.

9. Conclusions

Serverless computing has evolved significantly within modern cloud architecture, allowing developers the ability to focus only on application logic while relying on platforms to manage infrastructure and scaling.

 doi:10.20944/preprints202512.0219.v1

This review presents a clear understanding of the current state of the art of serverless architecture mentioned in the fifty peer-reviewed studies by organizing their findings into a taxonomy consisting of application domains, methods and techniques, data sources, limitations and future work. The review tells us how serverless systems are being applied in various domains such as machine learning, IoT, networking, and high-performance computing, etc. while also presenting the techniques and methods researchers have used to address its operational challenges.

Despite the advantages of serverless computing, it still faces constant limitations, such as cold start delays, state management complexity, platform dependency, limited debugging capabilities, and constraints on execution environments etc. These limitations influence the performance of serverless computing in certain scenerios and environments. However, the research on serverless computing continues as improvements in resource scheduling, adaptive optimization, workflow orchestration, etc. are being made.

By gathering existing papers and identifying open challenges, this review provides a foundation for future research aimed at improving the efficiency, flexibility, and applicability of serverless architectures across a wide range of computational domains and tells us that serverless computing is a rapidly evolving field with considerable room for further innovation.

References

1. De Silva, D.; Hewawasam, L. The Impact of Software Testing on Serverless Applications. *IEEE Access* **2024**, *12*, 51086–51099. https://doi.org/10.1109/ACCESS.2024.3384459.
2. USA.; Nellore, N.S. Optimizing Cost and Performance in Serverless Batch ProcessingA Comparative Analysis of AWS Step Functions vs. *Journal of Mathematical & Computer Applications* **2022**, *1*, 1–5. https://doi.org/10.47363/JMCA/2022(1)E160.
3. Kodakandla, N. Serverless Architectures: A Comparative Study of Performance, Scalability, and Cost in Cloud-native Applications **2021**. *5*.
4. Faculty of Computing Muhammad Ali Jinnah University (MAJU) Karachi, Pakistan.; Nadeem, M.U.; Raazi, S.M.K.u.R.; Faculty of Computing Mohammad Ali Jinnah University (MAJU) Karachi, Pakistan.; Mehboob, B.; Faculty of Computer Science and Information Technology Superior University Lahore, Pakistan.; Ali, S.M.; Malaysian Institute of Information Technology, Universiti Kuala Lumpur (UniKL MIIT) Kuala Lumpur, Malaysia.; Raza, S.; Faculty of Computing Mohammad Ali Jinnah University (MAJU) Karachi, Pakistan. Cost Analysis of Running Web Application in Cloud Monolith, Microservice and Serverless Architecture. *Journal of Independent Studies and Research Computing* **2024**, *22*. https://doi.org/10.31645/JISRC.24.22.2.7.
5. Gias, A.U.; Casale, G. COCOA: Cold Start Aware Capacity Planning for Function-as-a-Service Platforms, 2020. arXiv:2007.01222 [cs], https://doi.org/10.48550/arXiv.2007.01222.
6. Maissen, P.; Felber, P.; Kropf, P.; Schiavoni, V. FaaSdom: A Benchmark Suite for Serverless Computing. In Proceedings of the Proceedings of the 14th ACM International Conference on Distributed and Event-based Systems, 2020, pp. 73–84. arXiv:2006.03271 [cs], https://doi.org/10.1145/3401025.3401738.
7. Golec, M.; Ozturac, R.; Pooranian, Z.; Singh Gill, S.; Buyya, R. IFaaSBus: A Security- and Privacy-Based Lightweight Framework for Serverless Computing Using IoT and Machine Learning. *ResearchGate* **2021**. https://doi.org/10.1109/TII.2021.3095466.
8. Lin, C.; Khazaei, H. Modeling and Optimization of Performance and Cost of Serverless Applications. *IEEE Transactions on Parallel and Distributed Systems* **2021**, *32*, 615–632. https://doi.org/10.1109/TPDS.2020.3028841.
9. Lee, S.; Yoon, D.; Yeo, S.; Oh, S. Mitigating Cold Start Problem in Serverless Computing with Function Fusion. *Sensors* **2021**, *21*, 8416. Publisher: Multidisciplinary Digital Publishing Institute, https://doi.org/10.3390/s21248416.
10. Majewski, M.; Pawlik, M.; Malawski, M. Algorithms for scheduling scientific workflows on serverless architecture. In Proceedings of the 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2021, pp. 782–789. https://doi.org/10.1109/CCGrid51090.2021.00095.
11. Mahmoudi, N.; Khazaei, H. Performance Modeling of Serverless Computing Platforms. *IEEE Transactions on Cloud Computing* **2022**, *10*, 2834–2847. https://doi.org/10.1109/TCC.2020.3033373.

 doi:10.20944/preprints202512.0219.v1

12. Ouyang, R.; Wang, J.; Xu, H.; Chen, S.; Xiong, X.; Tolba, A.; Zhang, X. A Microservice and Serverless Architecture for Secure IoT System. *Sensors* **2023**, *23*, 4868. Publisher: Multidisciplinary Digital Publishing Institute, https://doi.org/10.3390/s23104868.
13. Bac, T.P.; Tran, M.N.; Kim, Y. Serverless Computing Approach for Deploying Machine Learning Applications in Edge Layer. In Proceedings of the 2022 International Conference on Information Networking (ICOIN), 2022, pp. 396–401. ISSN: 1976-7684, https://doi.org/10.1109/ICOIN53446.2022.9687209.
14. Chahal, D.; Ramesh, M.; Ojha, R.; Singhal, R. High Performance Serverless Architecture for Deep Learning Workflows. In Proceedings of the 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2021, pp. 790–796. https://doi.org/10.1109/CCGrid51090.2021.00096.
15. Assogba, K.; Arif, M.; Rafique, M.M.; Nikolopoulos, D.S. On Realizing Efficient Deep Learning Using Serverless Computing. In Proceedings of the 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2022, pp. 220–229. https://doi.org/10.1109/CCGrid54584.2022.00031.
16. Castagna, F.; Trombetta, A.; Landoni, M.; Andreon, S. A Serverless Architecture for Efficient and Scalable Monte Carlo Markov Chain Computation. In Proceedings of the Proceedings of the 2023 7th International Conference on Cloud and Big Data Computing, Manchester United Kingdom, 2023; pp. 68–73. https://doi.org/10.1145/3616131.3616141.
17. Bermbach, D.; Karakaya, A.S.; Buchholz, S. Using application knowledge to reduce cold starts in FaaS services. In Proceedings of the Proceedings of the 35th Annual ACM Symposium on Applied Computing, Brno Czech Republic, 2020; pp. 134–143. https://doi.org/10.1145/3341105.3373909.
18. Khan, D.; Subba, B.; Sharma, S. Minimizing Cold Start Times in Serverless Deployments. In Proceedings of the Proceedings of the 2022 Fourteenth International Conference on Contemporary Computing, New York, NY, USA, 2022; IC3-2022, pp. 156–161. https://doi.org/10.1145/3549206.3549234.
19. Nguyen, C.; Bhuyan, M.; Elmroth, E. Taming Cold Starts: Proactive Serverless Scheduling with Model Predictive Control, 2025. arXiv:2508.07640 [cs] version: 1, https://doi.org/10.48550/arXiv.2508.07640.
20. Lakhai, V.; Kuzmych, O.; Seniv, M. An improved method for increasing maintainability in terms of serverless architecture application. In Proceedings of the 2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT), Lviv, Ukraine, 2023; pp. 1–4. https://doi.org/10.1109/CSIT61576.2023.10324273.
21. Martins, H.; Araujo, F.; Da Cunha, P.R. Benchmarking Serverless Computing Platforms. *Journal of Grid Computing* **2020**, *18*, 691–709. https://doi.org/10.1007/s10723-020-09523-1.
22. Mahmoudi, N.; Khazaei, H. Performance Modeling of Metric-Based Serverless Computing Platforms. *IEEE Transactions on Cloud Computing* **2023**, *11*, 1899–1910. https://doi.org/10.1109/TCC.2022.3169619.
23. Pubglob. SERVERLESS ARCHITECTURES: A COMPARATIVE STUDY ON ENVIRONMENTAL IMPACT AND SUSTAINABILITY IN GREEN COMPUTING **2024**. Publisher: OSF, https://doi.org/10.17605/OSF.IO/T6H2W.
24. Oliver, E.; Thomas, E.; Price, L.; Musa, Z.; Esther, D. Serverless Architecture Patterns for Scalable APIs **2024**.
25. Kumar, G. The Evolution and Impact of Serverless Architectures in Modern Fintech Platforms. Technical report, 2024. https://doi.org/10.9790/5933-1504036579.
26. Padyana, U.K.; Rai, H.P.; Ogeti, P.; Fadnavis, N.S.; Patil, G.B. Server less Architectures in Cloud Computing: Evaluating Benefits and Drawbacks. *Innovative Research Thoughts* **2020**, *6*, 1–12. https://doi.org/10.36676/irt.v10.i3.1439.
27. Ekwe-Ekwe, N.; Amos, L. The State of FaaS: An Analysis of Public Functions-as-a-Service Providers. In Proceedings of the 2024 IEEE 17th International Conference on Cloud Computing (CLOUD), 2024, pp. 430–438. ISSN: 2159-6190, https://doi.org/10.1109/CLOUD62652.2024.00055.
28. Kanamugire, J.; Alhajri, F.; Swen, J. *Serverless Security and Privacy*; 2024. https://doi.org/10.13140/RG.2.2.34691.52006.
29. Abu, M.; Mallick, M.; Nath, R. Securing the Server-less Frontier: Challenges and Innovative Solutions in Network Security for Server- less Computing **2024**. *193*, 1–45.
30. Tummalachervu, C.K. EXPLORING SERVER-LESS COMPUTING FOR EFFICIENT RESOURCE MANAGEMENT IN CLOUD ARCHITECTURES **2024**. p. 77.
31. Cinar, B. The Rise of Serverless Architectures: Security Challenges and Best Practices. *Asian Journal of Research in Computer Science* **2023**, *16*, 194–210. https://doi.org/10.9734/ajrcos/2023/v16i4382.
32. Kathiriya, S.; Challa, N.; Devineni, S.K. Serverless Architecture in LLMs: Transforming the Financial Industry's AI Landscape. *International Journal of Science and Research (IJSR)* **2023**, *12*, 2131–2136. https://doi.org/10.21275/SR24302224043.

 doi:10.20944/preprints202512.0219.v1

33. Ahmed, N.; Hossain, M.; Shadul, S.; Rimi, N.; Sarkar, I. Server less Architecture: Optimizing Application Scalability and Cost Efficiency in Cloud Computing **2022**. *01*, 1366–1380.
34. Daraojimba, A.I.; Ogeawuchi, J.C.; Abayomi, A.A.; Agboola, O.A.; Ogbuefi, E. Systematic Review of Serverless Architectures and Business Process Optimization **2021**. *5*.
35. Hassan, H.B.; Barakat, S.A.; Sarhan, Q.I. Survey on serverless computing. *Journal of Cloud Computing* **2021**, *10*, 39. https://doi.org/10.1186/s13677-021-00253-7.
36. Kuhlenkamp, J.; Werner, S.; Tai, S. The Ifs and Buts of Less is More: A Serverless Computing Reality Check. In Proceedings of the 2020 IEEE International Conference on Cloud Engineering (IC2E), 2020, pp. 154–161. https://doi.org/10.1109/IC2E48712.2020.00023.
37. Przybylski, B.; Żuk, P.; Rzadca, K. Data-driven scheduling in serverless computing to reduce response time. In Proceedings of the 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2021, pp. 206–216. https://doi.org/10.1109/CCGrid51090.2021.00030.
38. Sharaf, S. Security Issues in Serverless Computing Architecture. *International Journal of Emerging Trends in Engineering Research* **2020**, *8*, 539–544. https://doi.org/10.30534/ijeter/2020/43822020.
39. Seth, D.; Chintale, P. Performance Benchmarking of Serverless Computing Platforms. *International Journal of Computer Trends and Technology* **2024**, *72*, 160–167. https://doi.org/10.14445/22312803/IJCTT-V72I6P121.
40. Fakinos, I.; Tzenetopoulos, A.; Masouros, D.; Xydis, S.; Soudris, D. Sequence Clock: A Dynamic Resource Orchestrator for Serverless Architectures. In Proceedings of the 2022 IEEE 15th International Conference on Cloud Computing (CLOUD), 2022, pp. 81–90. ISSN: 2159-6190, https://doi.org/10.1109/CLOUD55607.2022.00024.
41. Mampage, A.; Karunasekera, S.; Buyya, R. Deadline-aware Dynamic Resource Management in Serverless Computing Environments. In Proceedings of the 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2021, pp. 483–492. https://doi.org/10.1109/CCGrid51090.2021.00058.
42. Carpio, F.; Michalke, M.; Jukan, A. Engineering and Experimentally Benchmarking a Serverless Edge Computing System. In Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM), 2021, pp. 1–6. https://doi.org/10.1109/GLOBECOM46510.2021.9685235.
43. Sheshadri, K.R.; Lakshmi, J. QoS aware FaaS for Heterogeneous Edge-Cloud continuum. In Proceedings of the 2022 IEEE 15th International Conference on Cloud Computing (CLOUD), 2022, pp. 70–80. ISSN: 2159-6190, https://doi.org/10.1109/CLOUD55607.2022.00023.
44. Wen, J.; Chen, Z.; Zhao, J.; Sarro, F.; Ping, H.; Zhang, Y.; Wang, S.; Liu, X. SCOPE: Performance Testing for Serverless Computing. *ACM Trans. Softw. Eng. Methodol.* **2025**, *34*, 227:1–227:30. https://doi.org/10.1145/3717609.
45. Golec, M.; Walia, G.K.; Kumar, M.; Cuadrado, F.; Gill, S.S.; Uhlig, S. Cold Start Latency in Serverless Computing: A Systematic Review, Taxonomy, and Future Directions. *ACM Computing Surveys* **2025**, *57*, 1–36. arXiv:2310.08437 [cs], https://doi.org/10.1145/3700875.
46. Yussupov, V.; Soldani, J.; Breitenbücher, U.; Brogi, A.; Leymann, F. FaaSten your decisions: A classification framework and technology review of function-as-a-Service platforms. *Journal of Systems and Software* **2021**, *175*, 110906. https://doi.org/10.1016/j.jss.2021.110906.
47. Scheuner, J.; Leitner, P. Function-as-a-Service performance evaluation: A multivocal literature review. *Journal of Systems and Software* **2020**, *170*, 110708. https://doi.org/10.1016/j.jss.2020.110708.
48. Shojaee Rad, Z.; Ghobaei-Arani, M. Data pipeline approaches in serverless computing: a taxonomy, review, and research trends. *Journal of Big Data* **2024**, *11*, 82. https://doi.org/10.1186/s40537-024-00939-0.
49. Lannurien, V.; d'Orazio, L.; Barais, O.; Boukhobza, J. Serverless Cloud Computing: State of the Art and Challenges; 2023; pp. 275–316. https://doi.org/10.1007/978-3-031-26633-1_11.
50. Vahidinia, P.; Farahani, B.; Aliee, F.S. Cold Start in Serverless Computing: Current Trends and Mitigation Strategies. In Proceedings of the 2020 International Conference on Omni-layer Intelligent Systems (COINS), 2020, pp. 1–7. https://doi.org/10.1109/COINS49042.2020.9191377.
51. Taibi, D.; Ioini, N.E.; Pahl, C.; Niederkofler, J.R.S. Patterns for Serverless Functions (Function-as-a-Service): A Multivocal Literature Review. 2025, pp. 181–192.
52. Barnabas, A.; Johnson, J.; Mabel, E. The Future of Cloud: Exploring Cost- Effective Serverless Architecture **2025**.
53. Gaurav Samdani.; Kabita Paul.; Flavia Saldanha. Serverless architectures for agentic AI deployment. *World Journal of Advanced Engineering Technology and Sciences* **2022**, *7*, 320–333. https://doi.org/10.30574/wjaets.2022.7.2.0144.

54. Decker, J.; Kasprzak, P.; Kunkel, J.M. Performance Evaluation of Open-Source Serverless Platforms for Kubernetes. *Algorithms* **2022**, *15*, 234. https://doi.org/10.3390/a15070234.
55. Page, M.J.; McKenzie, J.E.; Bossuyt, P.M.; Boutron, I.; Hoffmann, T.C.; Mulrow, C.D.; Shamseer, L.; Tetzlaff, J.M.; Akl, E.A.; Brennan, S.E.; et al. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *BMJ* **2021**, *372*, n71. Publisher: British Medical Journal Publishing Group Section: Research Methods & Reporting, https://doi.org/10.1136/bmj.n71.
56. Gupta, V.; Kadhe, S.; Courtade, T.; Mahoney, M.W.; Ramchandran, K. OverSketched Newton: Fast Convex Optimization for Serverless Systems. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 288–297. https://doi.org/10.1109/BigData50022.2020.9378289.
57. Banaie, F.; Djemame, K.; Alhindi, A.; Kelefouras, V. Energy efficiency support for software defined networks: a serverless computing approach. *Future Generation Computer Systems* **2026**, *176*, 108121. https://doi.org/10.1016/j.future.2025.108121.
58. Tran, M.N.; Kim, Y. Design of 5G Architecture Enhancements for Supporting Serverless Computing. *ResearchGate* **2025**. https://doi.org/10.1109/ACCESS.2024.3490671.
59. Pakdil, M.E.; Çelik, R.N. Serverless Geospatial Data Processing Workflow System Design. *ISPRS International Journal of Geo-Information* **2022**, *11*, 20. Publisher: Multidisciplinary Digital Publishing Institute, https://doi.org/10.3390/ijgi11010020.
60. Suo, K.; Son, J.; Cheng, D.; Chen, W.; Baidya, S. Tackling Cold Start of Serverless Applications by Efficient and Adaptive Container Runtime Reusing. In Proceedings of the 2021 IEEE International Conference on Cluster Computing (CLUSTER), Portland, OR, USA, 2021; pp. 433–443. https://doi.org/10.1109/Cluster48925.2021.00018.
61. Kousiouris, G. A self-adaptive batch request aggregation pattern for improving resource management, response time and costs in microservice and serverless environments. In Proceedings of the 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC), 2021, pp. 1–10. ISSN: 2374-9628, https://doi.org/10.1109/IPCCC51483.2021.9679422.
62. Agarwal, S.; Rodriguez, M.A.; Buyya, R. A Reinforcement Learning Approach to Reduce Serverless Function Cold Start Frequency. In Proceedings of the 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Melbourne, Australia, 2021; pp. 797–803. https://doi.org/10.1109/CCGrid51090.2021.00097.
63. Gharibi, M.; Bhagavan, S.; Rao, P. FederatedTree: A Secure Serverless Algorithm for Federated Learning to Reduce Data Leakage. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 4078–4083. https://doi.org/10.1109/BigData52589.2021.9672039.
64. Jia, X.; Zhao, L. RAEF: Energy-efficient Resource Allocation through Energy Fungibility in Serverless. In Proceedings of the 2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS), 2021, pp. 434–441. ISSN: 2690-5965, https://doi.org/10.1109/ICPADS53394.2021.00060.
65. Zhao, Y.; Uta, A. Tiny Autoscalers for Tiny Workloads: Dynamic CPU Allocation for Serverless Functions. In Proceedings of the 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2022, pp. 170–179. https://doi.org/10.1109/CCGrid54584.2022.00026.
66. Govind, H.; González–Vélez, H. Benchmarking Serverless Workloads on Kubernetes. In Proceedings of the 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2021, pp. 704–712. https://doi.org/10.1109/CCGrid51090.2021.00085.
67. Shah, N.P. Design of a Reference Architecture for Serverless IoT Systems. In Proceedings of the 2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS), 2021, pp. 1–6. https://doi.org/10.1109/COINS51742.2021.9524180.
68. Jayaraman, S.; Reddy, C.; Khabiri, E.; Patel, D.; Bhamidipaty, A.; Kalagnanam, J. Asset Modeling using Serverless Computing. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 4084–4090. https://doi.org/10.1109/BigData52589.2021.9671711.
69. Byrne, A.; Nadgowda, S.; Coskun, A.K. ACE: Just-in-time Serverless Software Component Discovery Through Approximate Concrete Execution. In Proceedings of the Proceedings of the 2020 Sixth International Workshop on Serverless Computing, Delft Netherlands, 2020; pp. 37–42. https://doi.org/10.1145/3429880.3430098.
70. Safaryan, G.; Jindal, A.; Chadha, M.; Gerndt, M. SLAM: SLO-Aware Memory Optimization for Serverless Applications. In Proceedings of the 2022 IEEE 15th International Conference on Cloud Computing (CLOUD), 2022, pp. 30–39. ISSN: 2159-6190, https://doi.org/10.1109/CLOUD55607.2022.00019.

 doi:10.20944/preprints202512.0219.v1

71. Nishimiya, K.; Imai, Y. Serverless Architecture for Service Robot Management System. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 11379–11385. ISSN: 2577-087X, https://doi.org/10.1109/ICRA48506.2021.9561824.
72. Hanaforoosh, M.; Azgomi, M.A.; Ashtiani, M. Reducing the cost of cold start time in serverless function executions using granularity trees. *Future Generation Computer Systems* **2025**, *164*, 107604. https://doi.org/10.1016/j.future.2024.107604.
73. Joosen, A.; Hassan, A.; Asenov, M.; Singh, R.; Darlow, L.; Wang, J.; Deng, Q.; Barker, A. Serverless Cold Starts and Where to Find Them, 2024. arXiv:2410.06145 [cs], https://doi.org/10.48550/arXiv.2410.06145.
74. Tang, Y.; Yang, J. Lambdata: Optimizing Serverless Computing by Making Data Intents Explicit. In Proceedings of the 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), 2020, pp. 294–303. ISSN: 2159-6190, https://doi.org/10.1109/CLOUD49709.2020.00049.
75. Yu, H.; Irissappane, A.A.; Wang, H.; Lloyd, W.J. FaaSRank: Learning to Schedule Functions in Serverless Platforms. In Proceedings of the 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), 2021, pp. 31–40. https://doi.org/10.1109/ACSOS52086.2021.00023.